Video and Image Processing Example Design

© June 2011 AN-427-9.0

Introduction

The Altera® Video and Image Processing (VIP) Example Design demonstrates dynamic scaling and clipping of a standard definition video stream in either National Television System Committee (NTSC) or phase alternation line (PAL) format and picture-in-picture mixing with a background layer. The video stream is output in high definition resolution (1024×768) over a digital video interface (DVI).

The example design demonstrates a framework for rapid development of video and image processing systems using the parameterizable MegaCore® functions that are available in the Video and Image Processing Suite. Available functions are listed in Table 1.

Table 1. Video and Image Processing Suite MegaCore Functions

MegaCore Function	Description
Clocked Video Input	Converts clocked video formats to Avalon® Streaming (Avalon-ST) Video.
Clocked Video Output	Converts Avalon-ST Video to clocked video formats.
Frame Buffer	Buffers video frames into external RAM.
Color Plane Sequencer	Changes how color plane samples are transmitted across an Avalon-ST interface.
Clipper	Clips video streams.
Color Space Converter	Transforms video data between color spaces.
Gamma Corrector	Corrects video streams for the physical properties of display devices.
Chroma Resampler	Resamples video data to and from common sampling formats.
Deinterlacer II	Converts interlaced video to progressive video.
Alpha Blending Mixer	Mixes up to 12 input image layers.
2D FIR Filter	Performs 2D convolution using matrices of 3×3, 5×5, or 7×7 coefficients.
2D Median Filter	Applies 3×3, 5×5, or 7×7 pixel median filters to video images.
Scaler II	Resizes video streams.
Test Pattern Generator	Generates a test pattern video stream.
Frame Reader	Reads a video frame from external RAM.
Control Synchronizer	Synchronizes video stream changes with control port changes.
Switch	Switches video streams on frame boundaries at run time.
Interlacer	Converts progressive video to interlaced video.

For more information about these MegaCore functions, refer to the *Video and Image Processing Suite User Guide*.

These functions allow you to fully integrate common video functions with video interfaces, processors, and external memory controllers.

The example design uses an Altera Cyclone® III EP3C120 development board connected by high-speed mezzanine card (HSMC) interfaces to Bitec HSMC Quad Video and DVI daughtercards.

For information about the EP3C120 development board, refer to the *Cyclone III Development Board Reference Manual*.

A video source is input through an analog composite port on a Bitec HSMC Quad Video daughtercard, which generates a digital output in BT656 format.

A number of common video functions are performed on this input stream in the FPGA. These functions include clipping, chroma resampling, motion adaptive deinterlacing, color space conversion, picture-in-picture mixing, and polyphase scaling.

The input and output video interfaces on the two HSMC daughtercards (TVP5154 and TFP410 chips) are configured and initialized by software running on a Nios® II processor. Nios II software that demonstrates how to control the clocked video input, clocked video output, and mixer functions at run-time is also provided.

The video system is implemented using the Qsys system level design tool. This abstracted design tool provides an easy path to system integration of the video processing data path with a NTSC or PAL video input, DVI output, Nios II processor for configuration and control, and an external DDR2 memory controller.

The Video and Image Processing Suite MegaCore functions have common open Avalon-ST data interfaces and Avalon Memory-Mapped (Avalon-MM) control interfaces to facilitate connection of a chain of video functions and video system modeling. In addition, video data is transmitted between the Video and Image Processing Suite functions using the Avalon-ST Video protocol, which facilitates building run-time controllable systems and error recovery.

For a full description of how the Avalon-ST Video protocol and Avalon interfaces are implemented, refer to the *Interfaces* chapter in the *Video and Image Processing Suite User Guide*.

For more information about the Avalon-MM and Avalon-ST interfaces, refer to the *Avalon Interface Specifications.*

Qsys is a system development tool that allows you to create hardware and software system modules with a customized set of system peripherals. Qsys automatically creates the bus arbitration logic connecting the individual components together to create an overall system.

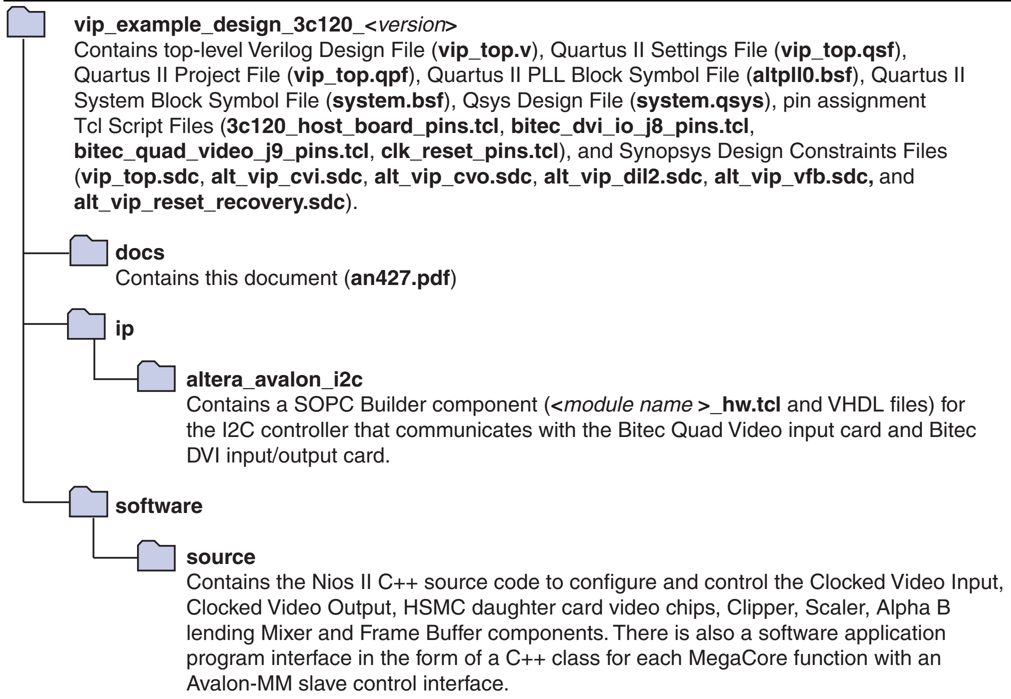
For more information about Qsys, refer to the *System Design with Qsys* section in volume 1 of the *Quartus II Handbook*.

Installing the Example Design

You can download the example design files in a **.zip** file from the Altera website. Figure 1 shows the directory structure for the example design files after you extract them from the **.zip** file.

The top-level directory **vip_example_design_3c120_<*version*>** contains a Quartus II project file (**vip_top.qpf**). You can open and explore the design as described in "Review the Example Design" on page 11.

Figure 1. Example Design Directory Structure *(1)*



vip_example_design_3c120_<*version*>
Contains top-level Verilog Design File (**vip_top.v**), Quartus II Settings File (**vip_top.qsf**), Quartus II Project File (**vip_top.qpf**), Quartus II PLL Block Symbol File (**altpll0.bsf**), Quartus II System Block Symbol File (**system.bsf**), Qsys Design File (**system.qsys**), pin assignment Tcl Script Files (**3c120_host_board_pins.tcl**, **bitec_dvi_io_j8_pins.tcl**, **bitec_quad_video_j9_pins.tcl**, **clk_reset_pins.tcl**), and Synopsys Design Constraints Files (**vip_top.sdc**, **alt_vip_cvi.sdc**, **alt_vip_cvo.sdc**, **alt_vip_dil2.sdc**, **alt_vip_vfb.sdc,** and **alt_vip_reset_recovery.sdc**).

docs
Contains this document (**an427.pdf**)

ip

altera_avalon_i2c
Contains a SOPC Builder component (<*module name* >**_hw.tcl** and VHDL files) for the I2C controller that communicates with the Bitec Quad Video input card and Bitec DVI input/output card.

software

source
Contains the Nios II C++ source code to configure and control the Clocked Video Input, Clocked Video Output, HSMC daughter card video chips, Clipper, Scaler, Alpha B lending Mixer and Frame Buffer components. There is also a software application program interface in the form of a C++ class for each MegaCore function with an Avalon-MM slave control interface.

Note to Figure 1:

(1) The **vip_example_design_3c120_**<*version*> directory also includes other files that are included for convenience but can be regenerated using the tools described in this application note.

System Requirements

This section describes the hardware and software requirements to run the video example design.

Hardware Requirements

The video and image processing example design requires the following hardware components:

- Cyclone III Video Development Kit including:
  - Cyclone III EP3C120 host development board
  - Bitec HSMC Quad Video input daughtercard
  - Bitec HSMC DVI input/output daughtercard
- Any NTSC video source with composite output (such as a DVD player or Apple iPod)
- A monitor or display with a DVI interface supporting 1024×768 resolution
- One DVI cable to connect the DVI output to the monitor

The example design has been verified using the following hardware:

- An Apple video iPod (An Apple TV out cable is required to connect the Apple Video iPod to the composite input.)
- A Sony Handycam HDR-CX150

Software Requirements

Ensure that you have extracted the example design **.zip** file and installed the software provided with the development kit.

For information about the software installation, refer to the documentation provided with the Cyclone III Video Development Kit.

You must have the following software installed on your PC:

- Quartus II software, version 11.0
- MegaCore IP Library, version 11.0 (installed with the Quartus II software)
- Nios II Embedded Design Suite, version 11.0

This application note assumes that you installed the software into the default locations.

Video Design Flow

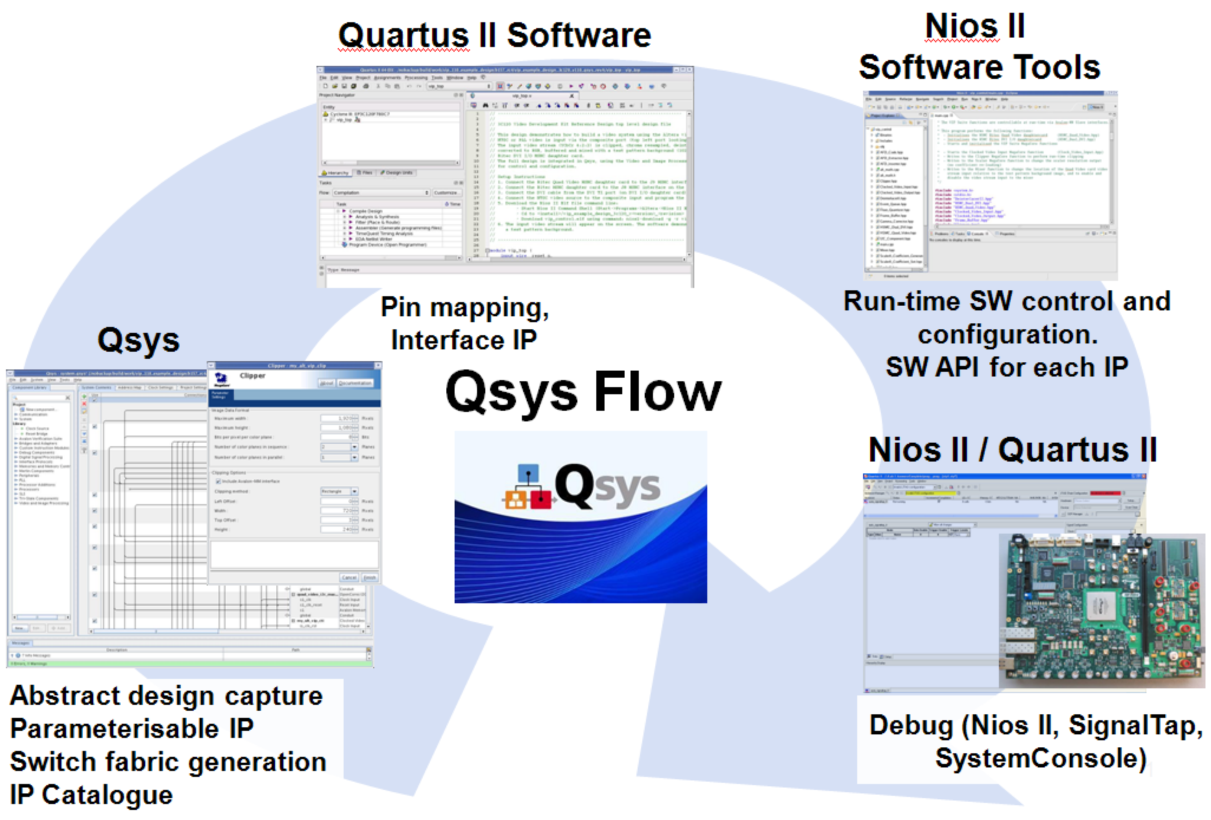
The video and image processing example design demonstrates a simple, yet highly parameterizable, design flow for rapid system development.

Figure 2 provides a high-level view of the design flow you typically experience within Altera's video design framework.

The video design framework provides the following features:

- Open interface and protocol standards to enable design reuse and connection of custom IP functions with off-the-shelf IP including:
  - Data streaming interfaces and protocols for transmission of video data between IP functions in the framework (Avalon-ST Video protocol layers on the Avalon-ST interface).
  - Control interfaces (Avalon-MM master and slave interfaces).
  - Random access to external memory (Avalon-MM master and slave interfaces).
- System level tools and design methodology for rapid system construction, integration, and redesign. Qsys takes advantage of standard interfaces by presenting an abstract view of the design and generating an application-specific switch fabric to construct the system.
- Parameterizable MegaCore IP functions that enable you to quickly construct complete video systems.
- Reference designs that demonstrate the capabilities of the video framework.
- Development kits to rapidly prototype the designs.

Figure 2. Qsys Design Flow



The example design described in this application note demonstrates each of these framework aspects. The video design tool flow is a key component that requires further description.

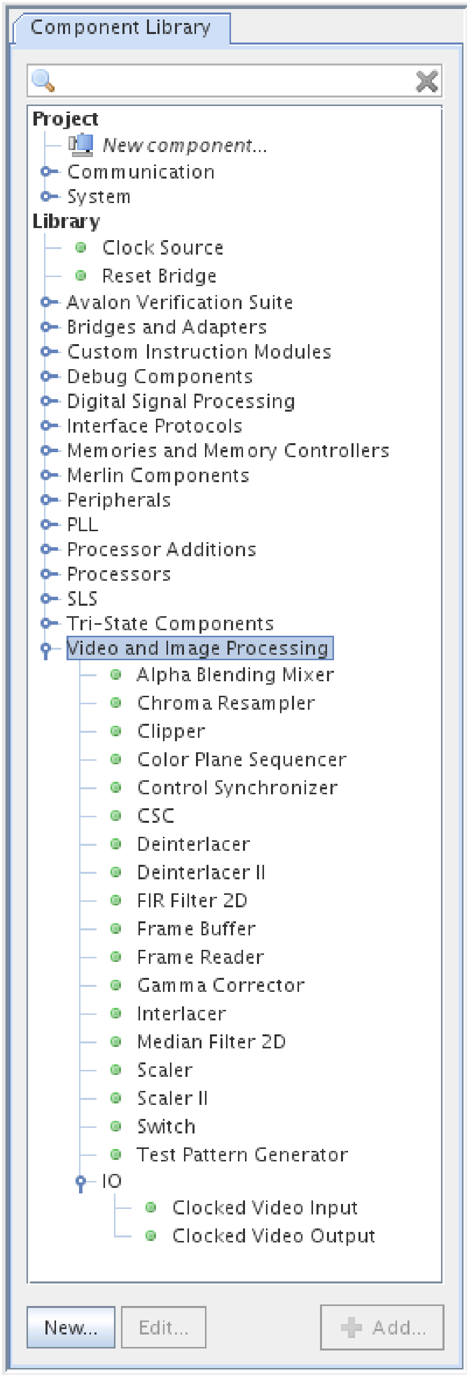
Qsys

The Qsys flow is the primary design flow for rapid video system development. Specifically, Qsys simplifies the process of system design, including the data path, processor core, and external memory integration. The tool enables you to capture the design at an abstract level, with single point-to-point data connections rather than connecting individual data and control interface wires.

All connections in the Qsys system use Avalon-ST and Avalon-MM interfaces. You can connect Video and Image Processing Suite MegaCore functions that support Avalon-ST Video protocol for data transmission with the click of a button. The MegaCore functions are displayed in the Qsys **Component Library** tab under **Video and Image Processing** as shown in Figure 3.

Qsys automatically generates an interconnect switch fabric, including arbitration logic to connect the memory mapped masters and slaves together. A common example is a system that contains multiple Avalon-MM masters that buffer video data in an external memory using a single memory controller.

Figure 3. Component Library Tab in Qsys



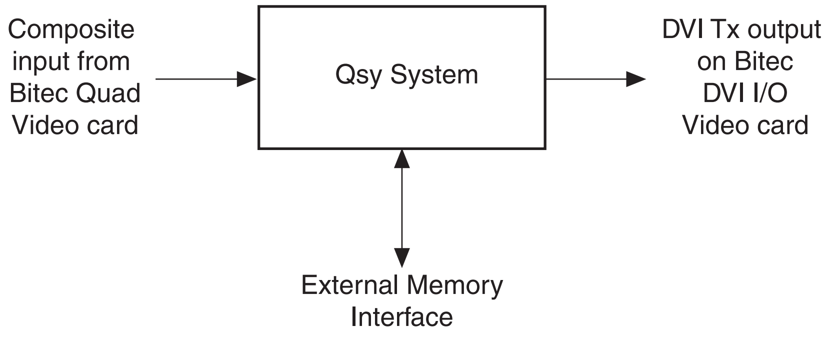
Component Library

Project
- *New component...*
- Communication
- System

Library
- Clock Source
- Reset Bridge
- Avalon Verification Suite
- Bridges and Adapters
- Custom Instruction Modules
- Debug Components
- Digital Signal Processing
- Interface Protocols
- Memories and Memory Controllers
- Merlin Components
- Peripherals
- PLL
- Processor Additions
- Processors
- SLS
- Tri-State Components
- Video and Image Processing
  - Alpha Blending Mixer
  - Chroma Resampler
  - Clipper
  - Color Plane Sequencer
  - Control Synchronizer
  - CSC
  - Deinterlacer
  - Deinterlacer II
  - FIR Filter 2D
  - Frame Buffer
  - Frame Reader
  - Gamma Corrector
  - Interlacer
  - Median Filter 2D
  - Scaler
  - Scaler II
  - Switch
  - Test Pattern Generator
  - IO
    - Clocked Video Input
    - Clocked Video Output

The Qsys system typically has three main external connections as shown in Figure 4.

Figure 4. External Connections to Qsys System


Composite input from Bitec Quad Video card
Qsy System
DVI Tx output on Bitec DVI I/O Video card
External Memory Interface

Video Input from an External Video Interface

The connection to the external video interface is made using a parameterizable Clocked Video Input MegaCore function. This function provides a bridge between a clocked video interface, such as a serial digital interface (SDI) MegaCore function, and the Avalon-ST Video flow controlled domain.

Video Output to an External Video Interface

The connection to the external video interface uses a parameterizable Clocked Video Output MegaCore function. This function provides a bridge between the Avalon-ST Video flow controlled domain and a clocked video interface (such as DVI).

Connection to an External Memory Interface

The connection to the external memory interface uses an Altera DDR2 SDRAM Controller MegaCore function. Qsys generates the application-specific switch fabric to arbitrate between multiple masters trying to access the controller.

For information about Qsys, refer to the Quartus II Help.

Quartus II Software

The top-level system is described within the Quartus II software environment and the Qsys system is integrated into the top-level design.

The Quartus II software environment is well suited for mapping the Qsys system external connections in the form of exported wires to video interface IP (such as SDI) and memory interfaces (such as DDR2) as well as making the appropriate pin assignments. A wide range of tools is provided to facilitate timing closure and perform hardware compilation to generate an FPGA programming file.

In addition, the Quartus II software provides the SignalTap™ II Logic Analyzer, a system-level debugging tool that captures and displays real-time signal behavior giving you the ability to observe interactions between hardware and software in system designs.

For information about the Quartus II software, refer to the Quartus II Help.

Nios II Software Build Tools for Eclipse

The Nios II Software Build Tools (SBT) for Eclipse is the primary software development tool for the Nios II family of embedded processors. You can perform all software development tasks within the Nios II SBT for Eclipse, including editing, building, and debugging programs. The Nios II SBT for Eclipse provides a consistent development platform that supports all Nios II processor systems.

You can configure video interfaces and control video processing functions in the Nios II SBT for Eclipse software. These features provide a very rapid development cycle for software control code changes, without requiring hardware recompilation. This environment provides you with all the standard software debug tools, including breakpoints, views of memory, variables, and registers, and single stepping.

This flow is further enhanced by the inclusion of C++ software classes that provide a software application programming interface (API) between the Nios II control code and the Video and Image Processing Suite MegaCore functions. The C++ classes contain member functions to provide easy control of the MegaCore functions and easy access to useful data flow information such as the number of frames that are dropped or repeated by a frame buffer in the data path.

For information about the Nios II SBT for Eclipse software, refer to the *Nios II Software Developer's Handbook*.

Functional Description

Figure 5 on page 9 shows a block diagram for the video and image processing example design.

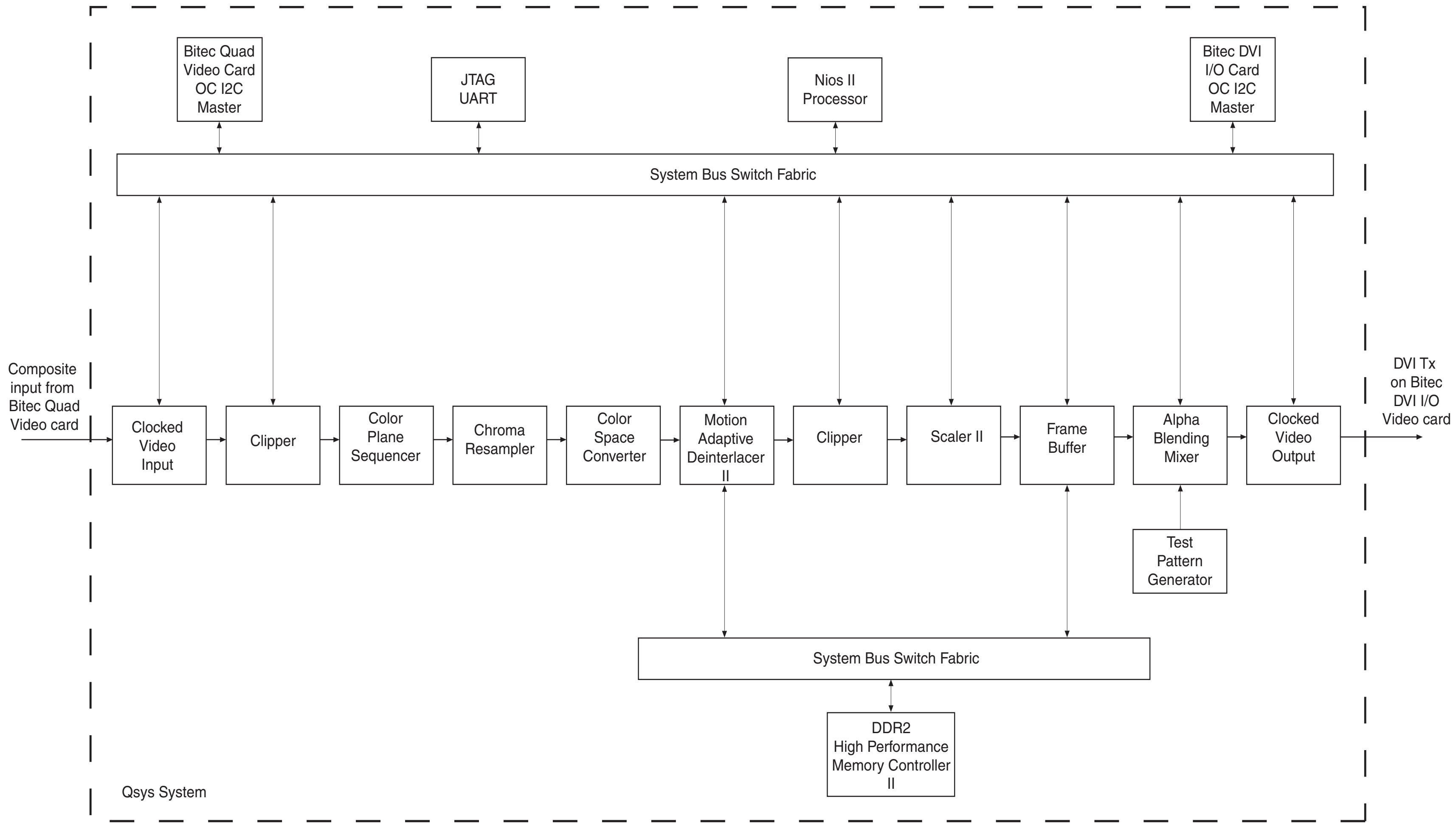
Video Input

An NTSC or PAL video stream (BT656 format) is input via the first composite input port (as shown in Figure 5) on the Bitec Quad Video daughtercard. The TVP5154 video decoder chip on the Bitec Quad Video daughtercard performs analog-to-digital conversion of the video input signals and must be configured before it can be used. An I2C interface is provided for this purpose.

The Nios II processor controls the configuration of the decoder chip by writing and reading data to and from the I2C bus controller, which performs writes and reads on the external bus as requested.

The video stream, with active picture data in YCbCr 4:2:2 format and associated embedded synchronization signals, is input from the daughtercard into the Clocked Video Input MegaCore function on the FPGA. This MegaCore function converts from a notionally clocked video format (such as BT656 or DVI) to the flow controlled Avalon-ST Video protocol.

Figure 5. Example Design Block Diagram



All Video and Image Processing Suite MegaCore functions (except the Line Buffer Compiler) transmit data using the Avalon-ST Video protocol, which increases productivity through design flexibility and reuse.

For details about the Avalon-ST Video protocol, refer to the to the *Interfaces* chapter in the *Video and Image Processing Suite User Guide*.

The Clocked Video Input function strips the incoming clocked video of horizontal and vertical blanking, leaving active picture data. Using this data with horizontal and vertical synchronization information, the Clocked Video Input function creates the necessary Avalon-ST Video control and active picture packets.

The active picture data is not converted; the color plane information remains the same as in the clocked video format.

The clocked video input also provides clock crossing capabilities to allow video formats running at different frequencies to enter the system. The clocked video input also provides greater design flexibility by enabling the video processing system clock to be decoupled from the video input pixel clock. In this example, the pixel clock is 27MHz and the system clock is 100MHz.

In addition to configuring the TVP5154 decoder chip, the Nios II processor starts the clocked video input.

Video Processing

The NTSC video stream input into the processing data path is 8 bits wide, in YCbCr 4:2:2 interlaced format, with each field input containing 720 pixels per line, and either 244 lines for each even field (f0) and 243 lines for each odd field (f1). The Luma (Y) and the Chroma (Cb and Cr) color planes are transmitted in sequence in the CbYCrY order.

The first processing function in the data path clips a rectangular region of 720 pixels by 240 lines from each field, offset three lines from the top of the field, and outputs fields with an equal number of odd and even lines, for further processing.

The clipped image video data is then converted from two colors in sequence (the Luma (Y) and Chroma (C) color planes) to two colors in parallel, using the Color Plane Sequencer MegaCore function. This operation demonstrates how you can change the format of the video data to suit the processing requirements of the functions in the data path and to satisfy the performance requirements of the application.

The Avalon-ST video data is chroma resampled to YCbCr 4:4:4 format prior to converting the color space to RGB from YCbCr. The interlaced video is then deinterlaced using a motion adaptive algorithm to produce a video stream in progressive format.

The chroma resampling is applied before the motion adaptive deinterlacing because the motion adaptive algorithm requires the input color channels to have the same sampling rate.

To allow run-time clipping every frame, a second clipper function is connected to the output from the deinterlacer. When the field resolution input to the motion adaptive deinterlacer changes, up to four fields might be required before the deinterlacer produces valid data. This delay is caused by the motion adaptive algorithm buffering two previous frames of the new video resolution to calculate the output pixel values.

The clipped progressive video stream is then scaled using the polyphase algorithm of the parameterizable scaling function (with four horizontal and four vertical taps).

To smooth out the burstiness of the video data flow, the frame buffer buffers the scaled video stream in external memory. The frame buffer also manages the rate change across the system video input and output by dropping and repeating video frames.

The video data output from the mixer is streamed into the clocked video output function, which supplies the DVI interface.

A DDR2 High Performance Memory Controller II buffers the video frames from both the frame buffer and the deinterlacing function in the video data path.

Video Output

A video stream (progressive RGB data) is output via the DVI Tx port on the Bitec HSMC DVI input/output daughtercard. The Nios II processor controls the configuration of the TFP410 DVI chip by writing or reading data to or from the I2C bus master component, which performs writes and reads on the external bus as requested.

The Clocked Video Output MegaCore function converts from the Avalon-ST Video protocol to clocked video formats (such as BT656 and DVI). The Clocked Video Output MegaCore function formats Avalon-ST Video into clocked video by inserting horizontal and vertical blanking and generating horizontal and vertical sync information using the Avalon-ST Video control and active picture packets. The active picture data is not converted; the color plane information remains the same as in the Avalon-ST Video format.

The clocked video output also provides clock crossing capabilities to provide greater design flexibility by enabling the video processing system clock to be decoupled from the video output pixel clock. In this example the output pixel clock is 65MHz and the system clock is 100MHz.

In addition, the clocked video output provides a queue where pixels can wait when the DVI output is in blanking and does not require pixel data. If this FIFO becomes full, then the flow controlled interface indicates that the clocked video output is not ready for data and earlier parts of the pipe are stalled.

Review the Example Design

This section describes how to construct a video processing application using the Altera video and image processing framework as demonstrated through the Video and Image Processing example design. This section includes a description of the top-level design file in the Quartus II software, the full hardware system in Qsys, and the configuration and control software code in the Nios II SBT for Eclipse.

Quartus II Top Level

To review the top-level Quartus II system, perform the following steps:

1. Launch the Quartus II software.
2. On the File menu, click **Open Project**, browse to <*example design install directory*>, and select the **vip_top.qpf** Quartus II project file.
3. On the File menu, click **Open**, browse to <*example design install directory*>, and select the **vip_top.v** top-level Verilog design file.

The Verilog design file also contains a text block which provides quick start instructions for setting up the example design.

The Verilog design file contains the following components:

- Clock source input into phase-locked loop (PLL) (`clock_source`)
- PLL megafunction to generate the Qsys system clock and DVI output pixel clocks (`altpll0`)
- Input pins for composite input channel 1 (from Bitec Quad Video input card on the HSMC interface J9) (`BITEC_QV_`)
- Qsys system containing video processing functions (`system`)
- DDR2 output pins (`ddr2_bot_`)
- Output pins `system` driving DVI output on Bitec HSMC DVI input/output daughtercard (J8) (`BITEC_DVI_IO_OUT_`)

The Quartus II project also contains pin assignment information for additional interfaces on the Quad Video input daughtercard and DVI input/output daughtercard.

The following Tcl files provide full pin assignments for reuse:

- **bitec_quad_video_j9_pins.tcl**
- **bitec_dvi_io_j8_pins.tcl**
- **3c120_host_board_pins.tcl** (includes memory assignments)
- **clk_reset_pins.tcl**

The following system timing constraints files are provided:

- **vip_top.sdc** - contains top-level clock requirements
- **alt_vip_cvi.sdc** - contains timing constraints for the Clocked Video Input MegaCore function
- **alt_vip_cvo.sdc** - contains timing constraints for Clocked Video Output MegaCore function
- **alt_vip_dil2.sdc** - contains timing constraints for the Deinterlacer II
- **alt_vip_reset_recovery.sdc** - contains timing constraints for recovery violations
- **alt_vip_vfb.sdc** - contains timing constraints for the Frame Buffer

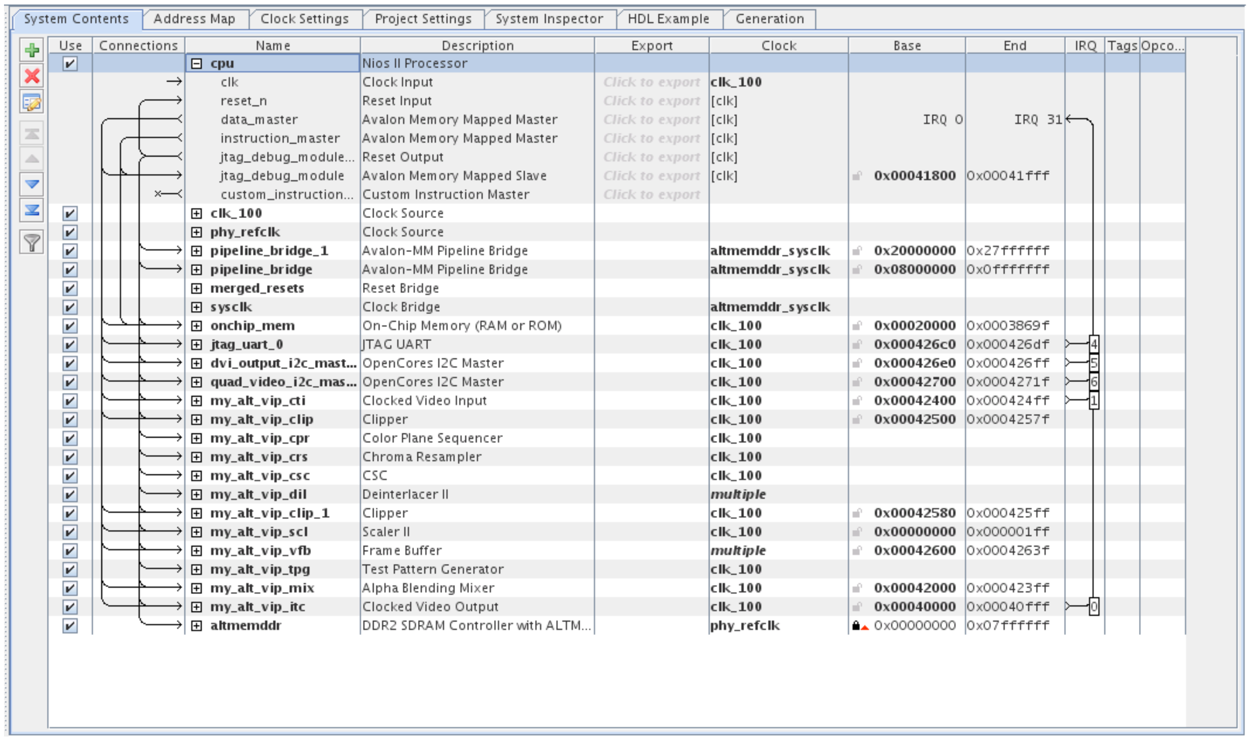
Qsys System

This section describes how to open the video and image processing example design in Qsys. To launch and review the Qsys system, with the **vip_top.qpf** project open in the Quartus II software, click **Qsys** on the Tools menu.

When you successfully launched Qsys, a dialog box is displayed for you to select the design file. Always select the **system.qsys** file for this design. Loading time for the design file might vary depending on your system.

Figure 6 shows the complete Qsys design.

Figure 6. Video and Image Processing Example Design in Qsys



Use	Connections	Name	Description	Export	Clock	Base	End	IRQ	Tags	Opco...
✓		cpu	Nios II Processor							
		clk	Clock Input	Click to export	clk_100					
		reset_n	Reset Input	Click to export	[clk]					
		data_master	Avalon Memory Mapped Master	Click to export	[clk]	IRQ 0	IRQ 31			
		instruction_master	Avalon Memory Mapped Master	Click to export	[clk]					
		jtag_debug_module...	Reset Output	Click to export	[clk]					
		jtag_debug_module	Avalon Memory Mapped Slave	Click to export	[clk]	0x00041800	0x00041fff			
		custom_instruction...	Custom Instruction Master	Click to export						
✓		clk_100	Clock Source							
✓		phy_refclk	Clock Source							
✓		pipeline_bridge_1	Avalon-MM Pipeline Bridge		altmemddr_sysclk	0x20000000	0x27ffffff			
✓		pipeline_bridge	Avalon-MM Pipeline Bridge		altmemddr_sysclk	0x08000000	0x0fffffff			
✓		merged_resets	Reset Bridge							
✓		sysclk	Clock Bridge		altmemddr_sysclk					
✓		onchip_mem	On-Chip Memory (RAM or ROM)		clk_100	0x00020000	0x0003869f			
✓		jtag_uart_0	JTAG UART		clk_100	0x000426c0	0x000426df	4		
✓		dvi_output_i2c_mast...	OpenCores I2C Master		clk_100	0x000426e0	0x000426ff	5		
✓		quad_video_i2c_mas...	OpenCores I2C Master		clk_100	0x00042700	0x0004271f	6		
✓		my_alt_vip_cti	Clocked Video Input		clk_100	0x00042400	0x000424ff	1		
✓		my_alt_vip_clip	Clipper		clk_100	0x00042500	0x0004257f			
✓		my_alt_vip_cpr	Color Plane Sequencer		clk_100					
✓		my_alt_vip_crs	Chroma Resampler		clk_100					
✓		my_alt_vip_csc	CSC		clk_100					
✓		my_alt_vip_dil	Deinterlacer II		*multiple*					
✓		my_alt_vip_clip_1	Clipper		clk_100	0x00042580	0x000425ff			
✓		my_alt_vip_scl	Scaler II		clk_100	0x00000000	0x000001ff			
✓		my_alt_vip_vfb	Frame Buffer		*multiple*	0x00042600	0x0004263f			
✓		my_alt_vip_tpg	Test Pattern Generator		clk_100					
✓		my_alt_vip_mix	Alpha Blending Mixer		clk_100	0x00042000	0x000423ff			
✓		my_alt_vip_itc	Clocked Video Output		clk_100	0x00040000	0x00040fff	0		
✓		altmemddr	DDR2 SDRAM Controller with ALTM...		phy_refclk	0x00000000	0x07ffffff			

The Qsys system contains the following key components:

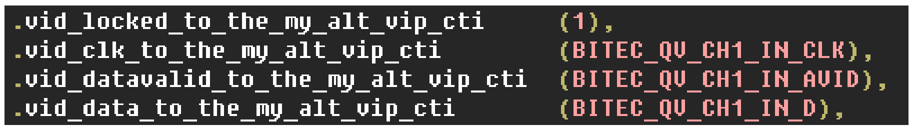
- Video processing data path (for video processing, system integration and buffering data)
- I2C bus masters (for configuration of video input/output chips)
- Bridge components (for design flexibility and performance)
- DDR2 external memory controller (for video buffering)
- Nios II processor and on-chip memory for program code (for system configuration and control)

Video Processing Data Path

The video processing data path is a chain of parameterizable video processing, integration, and buffering functions from the Video and Image Processing Suite. The video stream, in BT656 format, is input into the system through the Clocked Video Input MegaCore function. The Clocked Video Input forms the boundary of the Qsys design at the input, with the clocked video signals exported to the top-level system for connection to the input pins from the Quad Video daughtercard.

These exported signals are shown in system instantiation in the **vip_top.v** file (Figure 7).

Figure 7. Signals Exported from the Quad Video Daughtercard



```
.vid_locked_to_the_my_alt_vip_cti     (1),
.vid_clk_to_the_my_alt_vip_cti        (BITEC_QV_CH1_IN_CLK),
.vid_datavalid_to_the_my_alt_vip_cti  (BITEC_QV_CH1_IN_AVID),
.vid_data_to_the_my_alt_vip_cti       (BITEC_QV_CH1_IN_D),
```

1. Double-click **my_alt_vip_cti** in Qsys to display the MegaWizard™ interface for the Clocked Video Input, as shown in Figure 8.

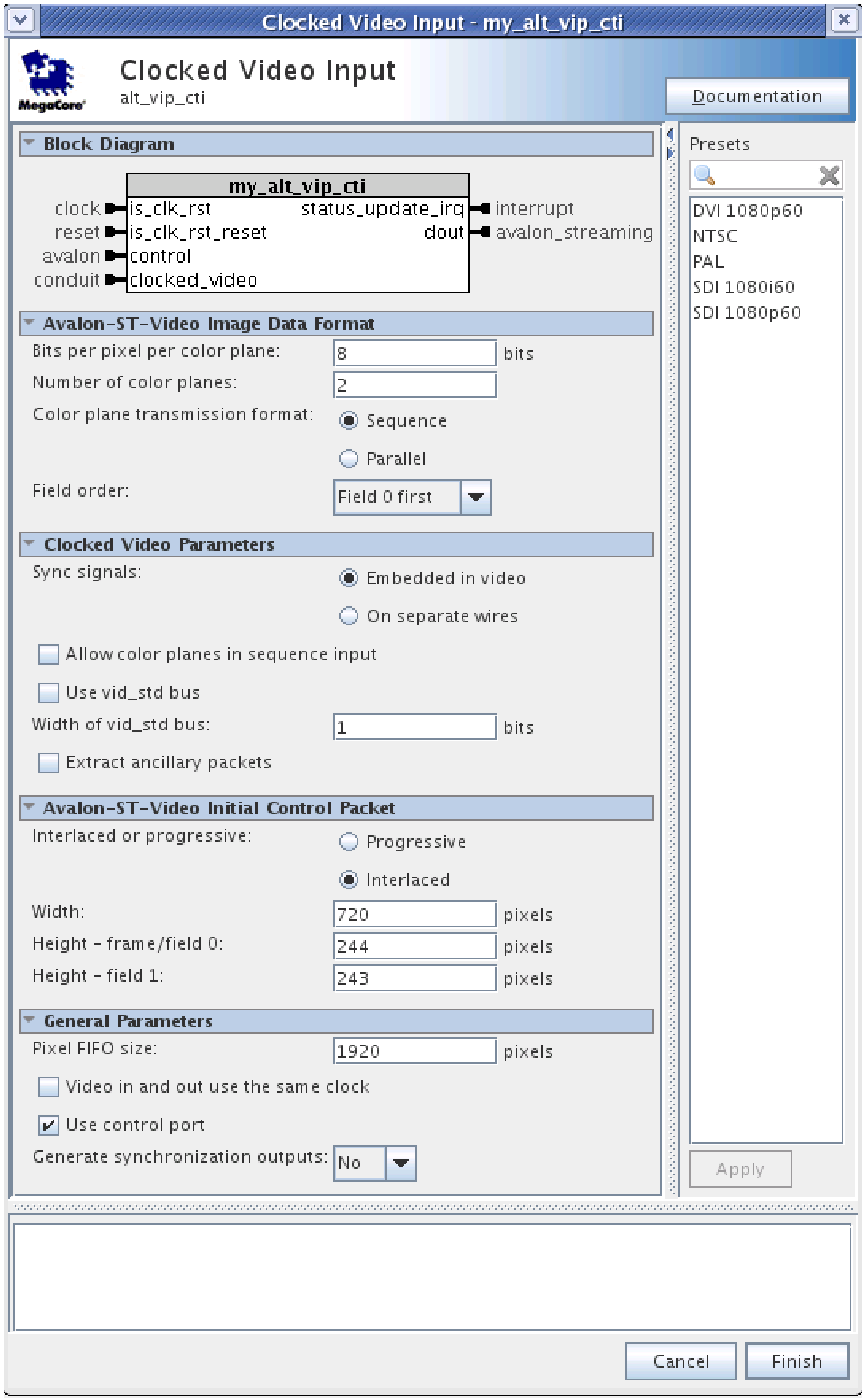
   The Clocked Video Input converts from a notionally clocked video format (BT656 in this example) to the flow-controlled Avalon-ST Video protocol. All Video and Image Processing Suite MegaCore functions (except the Line Buffer Compiler) transmit data using the Avalon-ST Video protocol, which increases productivity through design flexibility and reuse.

   For details about the Avalon-ST Video protocol, refer to the to the *Interfaces* chapter in the *Video and Image Processing Suite User Guide*.

   The Clocked Video Input strips the incoming clocked video of horizontal and vertical blanking, leaving active picture data, and using this data with the horizontal and vertical sync information creates the necessary Avalon-ST Video control and active picture packets. In this example, the clocked video input function is configured to accept video data with two color planes in sequence, namely, C-Y. The subsampled Cb and Cr color planes are treated as the C color plane.

   The Clocked Video Input can compute the resolution of a video frame only after it has received a full frame of data. Before the first video frame is output, a default command packet is output as specified under **Avalon-ST-Video Initial Control Packet** in the MegaWizard interface. This packet matches the resolution and format of the expected NTSC video input.

Figure 8. Parameter Settings for the Clocked Video Input



Presets: DVI 1080p60, NTSC, PAL, SDI 1080i60, SDI 1080p60

Avalon-ST-Video Image Data Format

Bits per pixel per color plane: 8 bits

Number of color planes: 2

Color plane transmission format: [✓ Sequence] Parallel

Field order: Field 0 first

Clocked Video Parameters

Sync signals: [✓ Embedded in video] On separate wires

☐ Allow color planes in sequence input

☐ Use vid_std bus

Width of vid_std bus: 1 bits

☐ Extract ancillary packets

Avalon-ST-Video Initial Control Packet

Interlaced or progressive: Progressive [✓ Interlaced]

Width: 720 pixels

Height - frame/field 0: 244 pixels

Height - field 1: 243 pixels

General Parameters

Pixel FIFO size: 1920 pixels

☐ Video in and out use the same clock

☑ Use control port

Generate synchronization outputs: No

The **Video in and out use the same clock** option is turned off, which configures the Clocked Video Input block to use separate clocks for the clocked video side and the flow controlled domain.

The **Use control port** option is turned on and provides an Avalon-MM slave port. In this system, the Nios II processor data master is connected to the Avalon-MM slave. This port gives the Nios II processor control when the clocked video input starts accepting data. The register map contains many status registers that provide feedback on the format of video entering the system (resolution, interlaced, or progressive). A status interrupt can also be used, but is not shown in this example.

For further information about the control register map, refer to *Appendix A* in the *Video and Image Processing Suite User Guide*.

2. The first processing function after the Clocked Video Input in the data path is a Clipper MegaCore function, which is clocked in the system clock domain (100 MHz). Double-click **my_alt_vip_clip** in Qsys to display the MegaWizard interface **Parameter Settings** page for the Clipper (Figure 9).

   The Clipper automatically reacts to changes in the image resolution at run-time up to a maximum width and height specified in the MegaWizard interface.

   You can configure the Clipper to include an Avalon-MM slave interface that enables run-time changes to the clipping region. In this example design, the software executed by the Nios II processor configures the Clipper to clip a rectangular region of 720 pixels by 240 lines from each field, offset three lines from the top of the field, outputting fields with an equal number of odd and even lines, for further processing.

Figure 9. Parameter Settings for the Clipper

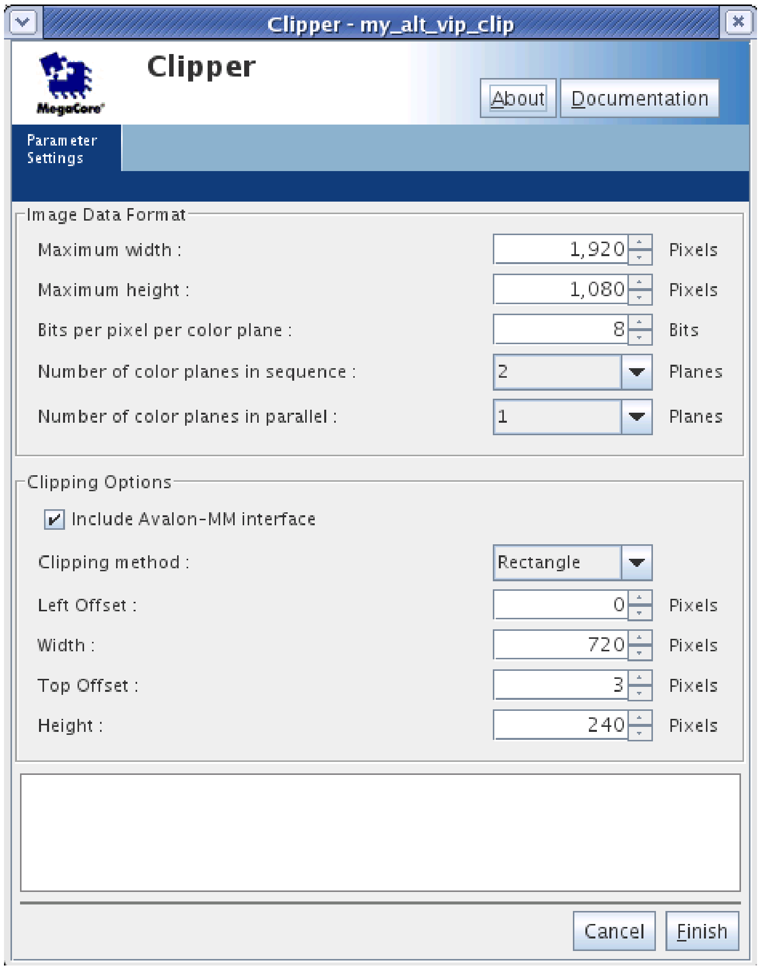


Image Data Format

Maximum width :	1,920	Pixels
Maximum height :	1,080	Pixels
Bits per pixel per color plane :	8	Bits
Number of color planes in sequence :	2	Planes
Number of color planes in parallel :	1	Planes

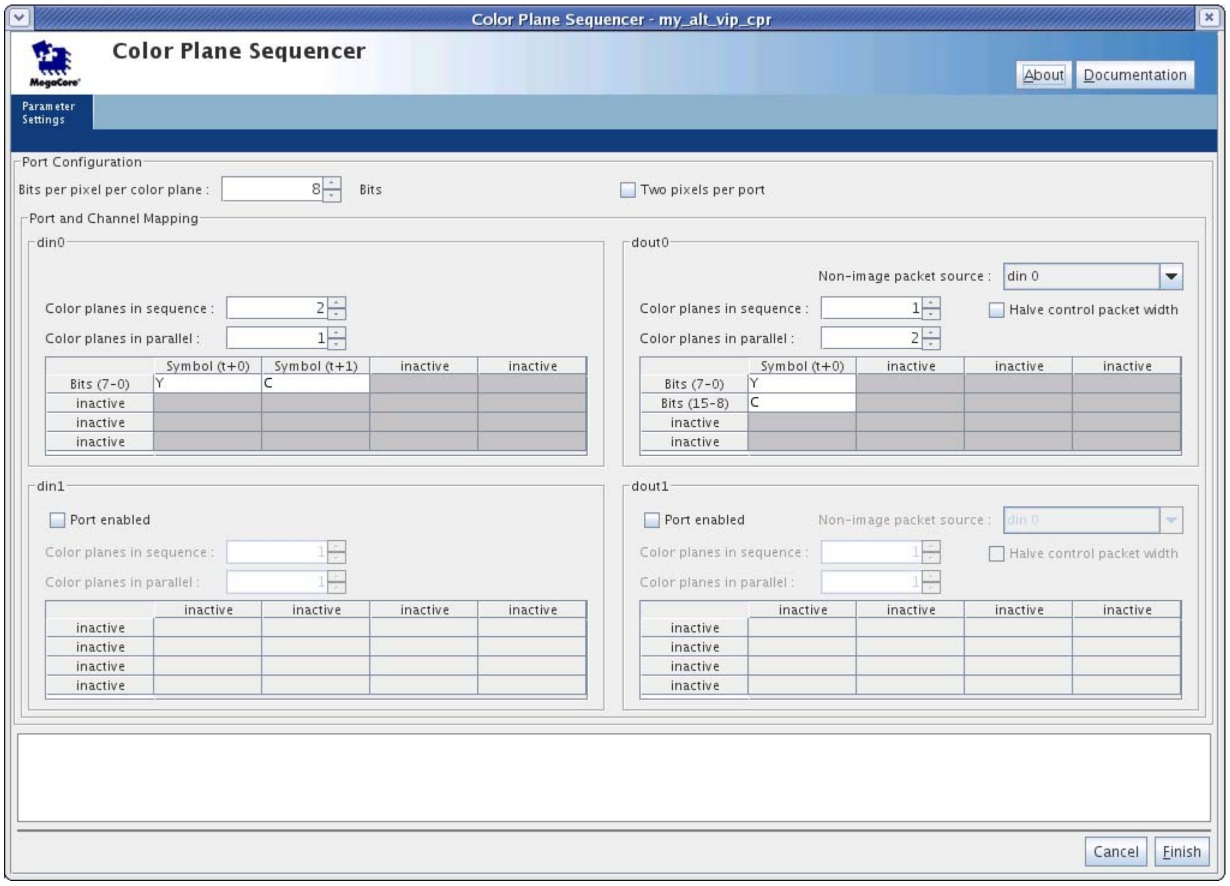
Clipping Options

✓ Include Avalon-MM interface

Clipping method :	Rectangle	
Left Offset :	0	Pixels
Width :	720	Pixels
Top Offset :	3	Pixels
Height :	240	Pixels

3. The clipped image video data is then converted from two colors in sequence (the Luma Y, and Chroma C, color planes) to two colors in parallel, using the Color Plane Sequencer MegaCore function. Double-click **my_alt_vip_cpr** in Qsys to display the MegaWizard interface **Parameter Settings** page for the Color Plane Sequencer (Figure 10).

Figure 10. Parameter Settings for the Color Plane Sequencer



Color Plane Sequencer - my_alt_vip_cpr

Color Plane Sequencer

Port Configuration

Bits per pixel per color plane : 8 Bits

Two pixels per port: unchecked

Port and Channel Mapping

din0

Color planes in sequence : 2

Color planes in parallel : 1

	Symbol (t+0)	Symbol (t+1)	inactive	inactive
Bits (7-0)	Y	C		
inactive				
inactive				
inactive				

dout0

Non-image packet source : din 0

Color planes in sequence : 1

Halve control packet width: unchecked

Color planes in parallel : 2

	Symbol (t+0)	inactive	inactive	inactive
Bits (7-0)	Y			
Bits (15-8)	C			
inactive				
inactive				

din1

Port enabled: unchecked

Color planes in sequence : 1

Color planes in parallel : 1

dout1

Port enabled: unchecked

Non-image packet source : din 0

Color planes in sequence : 1

Halve control packet width: unchecked

Color planes in parallel : 1

This function demonstrates how you can change the format of the video data to suit the processing requirements of the functions in the data path and to satisfy the performance requirements of the application.

4. The Avalon-ST Video data is then chroma resampled to YCbCr 4:4:4 format using the Chroma Resampler MegaCore function. Double-click **my_alt_vip_crs** in Qsys to display the MegaWizard interface **Parameter Settings** page for the Chroma Resampler (Figure 11).

Figure 11. Parameter Settings for the Chroma Resampler

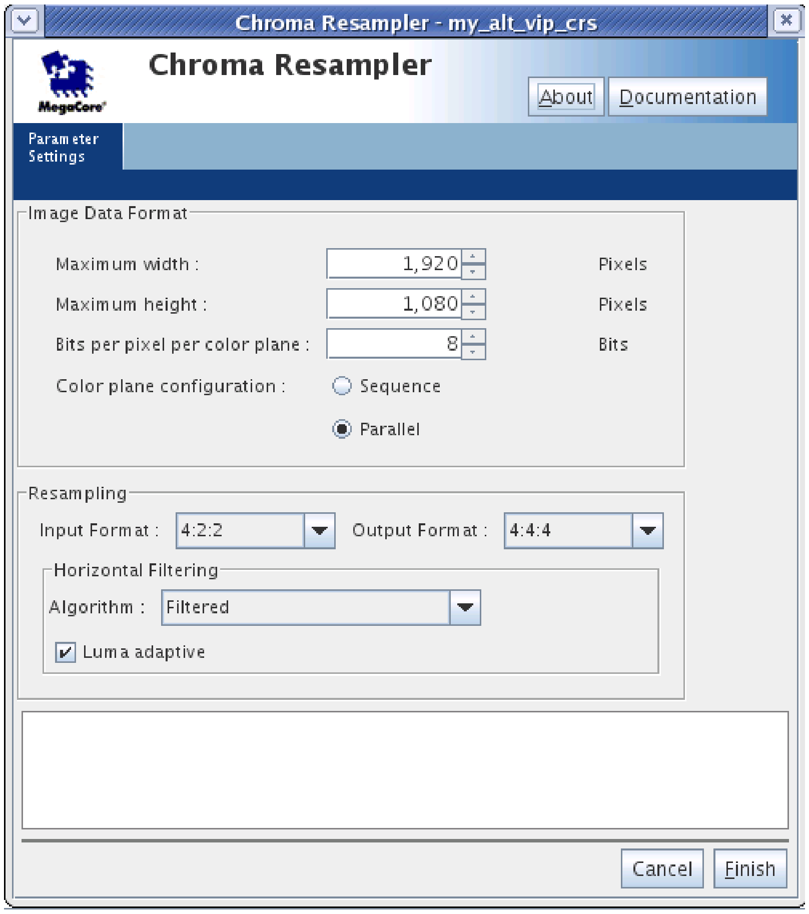


Image Data Format

Maximum width : 1,920 Pixels

Maximum height : 1,080 Pixels

Bits per pixel per color plane : 8 Bits

Color plane configuration : Sequence [✓ Parallel]

Resampling

Input Format : 4:2:2 Output Format : 4:4:4

Horizontal Filtering

Algorithm : Filtered

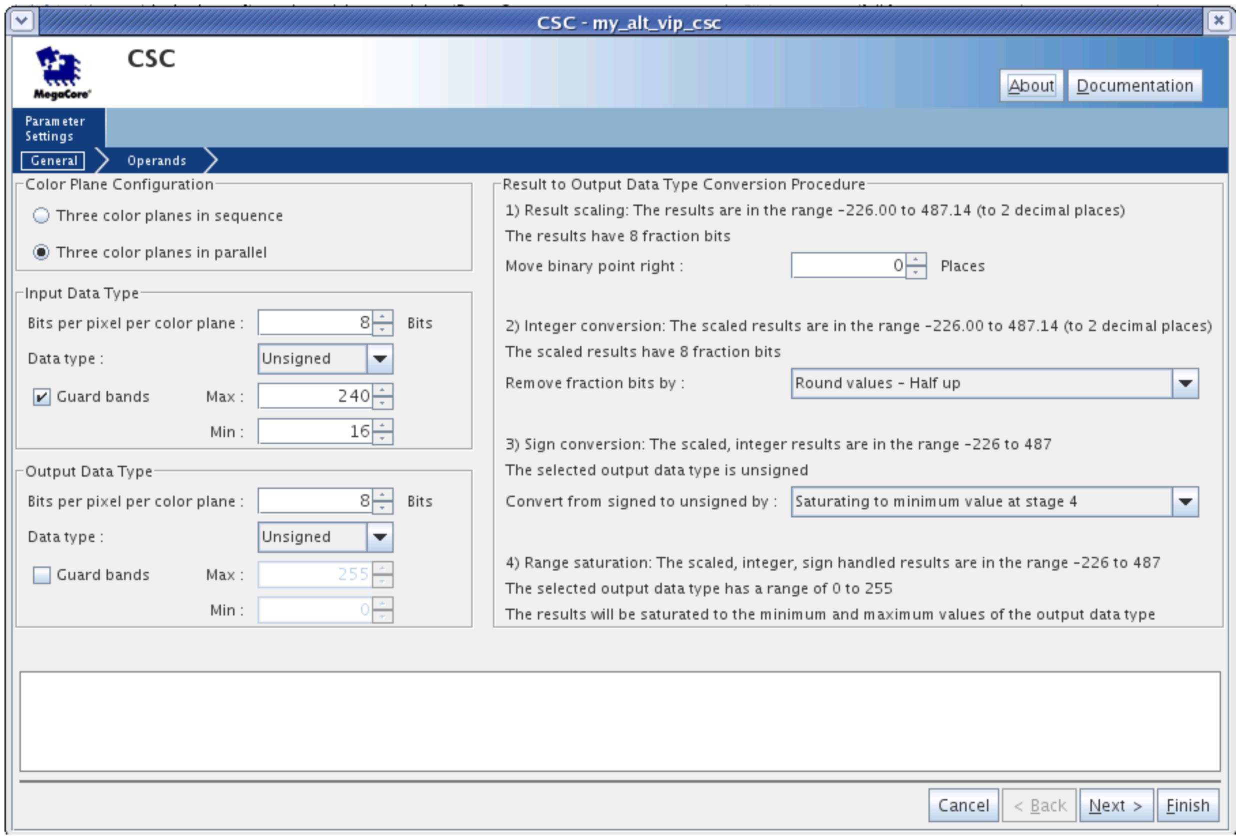
[✓ Luma adaptive]

In this example design, the Chroma Resampler is configured to support video data input up to a maximum resolution of 1920×1080 pixels. The Chroma Resampler converts the YCbCr 4:2:2 subsampled pixel stream to a 4:4:4 sampled stream using a filtered luma adaptive algorithm.

Vertical interpolation is not applicable in this example because 4:2:2 subsamples horizontally only.

5. The example design converts the YCbCr color space to the RGB color space using the Color Space Converter MegaCore function. Double-click **my_alt_vip_csc** in Qsys to display the MegaWizard interface **General Parameter Settings** page for the Color Space Converter (Figure 12).

Figure 12. General Parameter Settings for the Color Space Converter



Color Plane Configuration
( ) Three color planes in sequence
(●) Three color planes in parallel

Input Data Type
Bits per pixel per color plane : 8 Bits
Data type : Unsigned
[✓] Guard bands Max : 240
Min : 16

Output Data Type
Bits per pixel per color plane : 8 Bits
Data type : Unsigned
[ ] Guard bands Max : 255
Min : 0

Result to Output Data Type Conversion Procedure
1) Result scaling: The results are in the range -226.00 to 487.14 (to 2 decimal places)
The results have 8 fraction bits
Move binary point right : 0 Places

2) Integer conversion: The scaled results are in the range -226.00 to 487.14 (to 2 decimal places)
The scaled results have 8 fraction bits
Remove fraction bits by : Round values - Half up

3) Sign conversion: The scaled, integer results are in the range -226 to 487
The selected output data type is unsigned
Convert from signed to unsigned by : Saturating to minimum value at stage 4

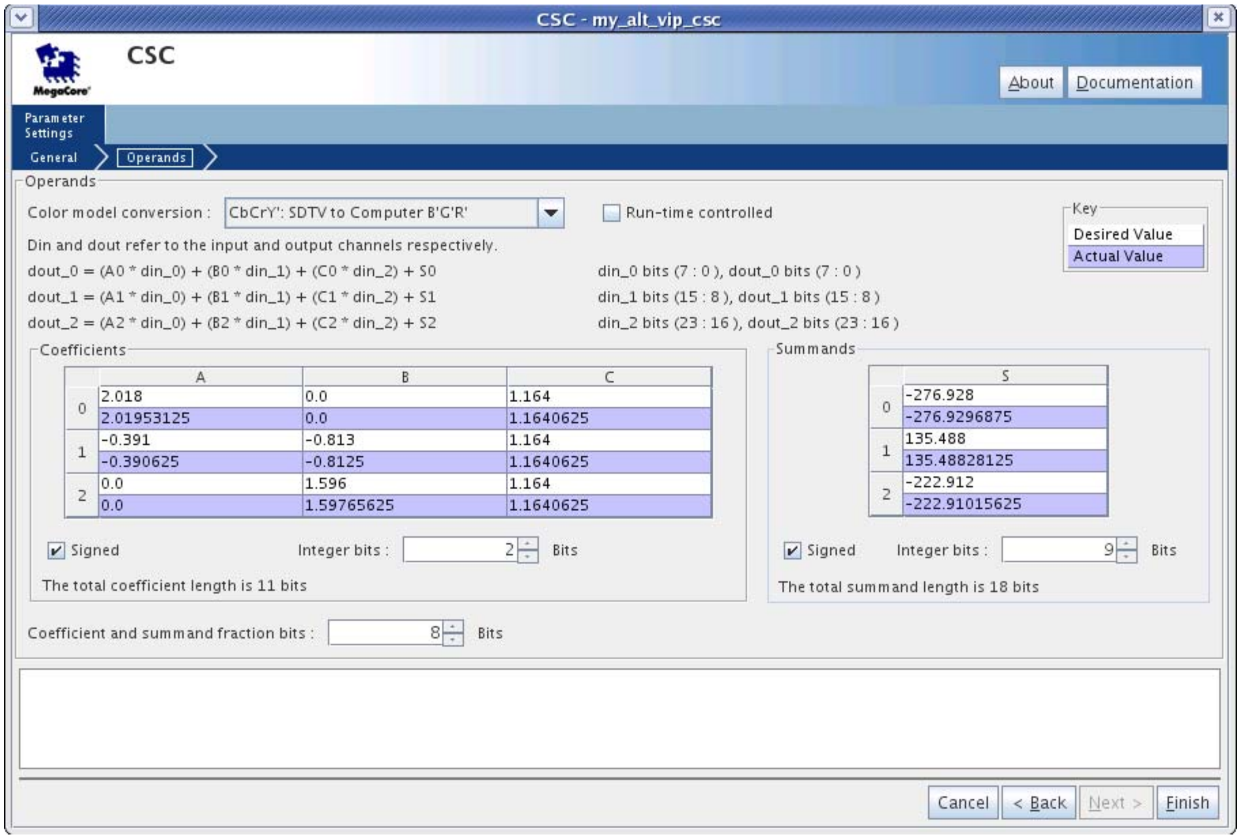
4) Range saturation: The scaled, integer, sign handled results are in the range -226 to 487
The selected output data type has a range of 0 to 255
The results will be saturated to the minimum and maximum values of the output data type

6. Click the **Operands** tab to display the coefficients used to convert between the YCbCr: SDTV and Computer RGB color spaces (Figure 13).

   In the RGB color space, the interlaced video is deinterlaced to produce a video stream in progressive format.

   The chroma resampling is applied before the motion adaptive deinterlacing because the motion adaptive algorithm requires the input color channels to have the same sampling rate.

Figure 13. Operands Parameter Settings for the Color Space Converter



Operands

Color model conversion : CbCrY': SDTV to Computer B'G'R'

Run-time controlled

Key: Desired Value / Actual Value

Din and dout refer to the input and output channels respectively.

```
dout_0 = (A0 * din_0) + (B0 * din_1) + (C0 * din_2) + S0     din_0 bits (7 : 0 ), dout_0 bits (7 : 0 )
dout_1 = (A1 * din_0) + (B1 * din_1) + (C1 * din_2) + S1     din_1 bits (15 : 8 ), dout_1 bits (15 : 8 )
dout_2 = (A2 * din_0) + (B2 * din_1) + (C2 * din_2) + S2     din_2 bits (23 : 16 ), dout_2 bits (23 : 16 )
```

Coefficients

	A	B	C
0	2.018	0.0	1.164
	2.01953125	0.0	1.1640625
1	-0.391	-0.813	1.164
	-0.390625	-0.8125	1.1640625
2	0.0	1.596	1.164
	0.0	1.59765625	1.1640625

✓ Signed Integer bits : 2 Bits

The total coefficient length is 11 bits

Summands

	S
0	-276.928
	-276.9296875
1	135.488
	135.48828125
2	-222.912
	-222.91015625

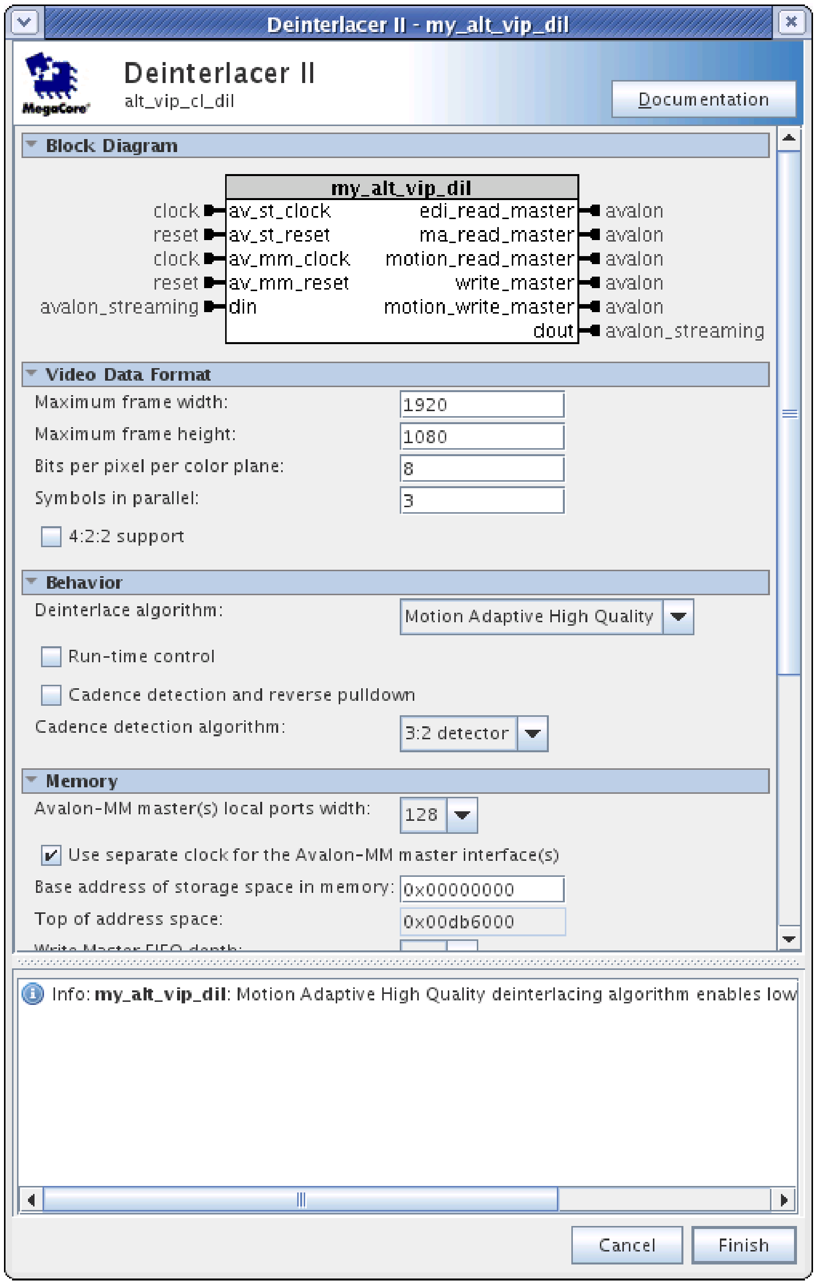
✓ Signed Integer bits : 9 Bits

The total summand length is 18 bits

Coefficient and summand fraction bits : 8 Bits

7. Double-click **my_alt_vip_dil** in Qsys to display the MegaWizard interface **Parameter Settings** page for the Deinterlacer II (Figure 14).

Figure 14. Parameter Settings for the Deinterlacer II



Maximum frame width: 1920
Maximum frame height: 1080
Bits per pixel per color plane: 8
Symbols in parallel: 3
☐ 4:2:2 support

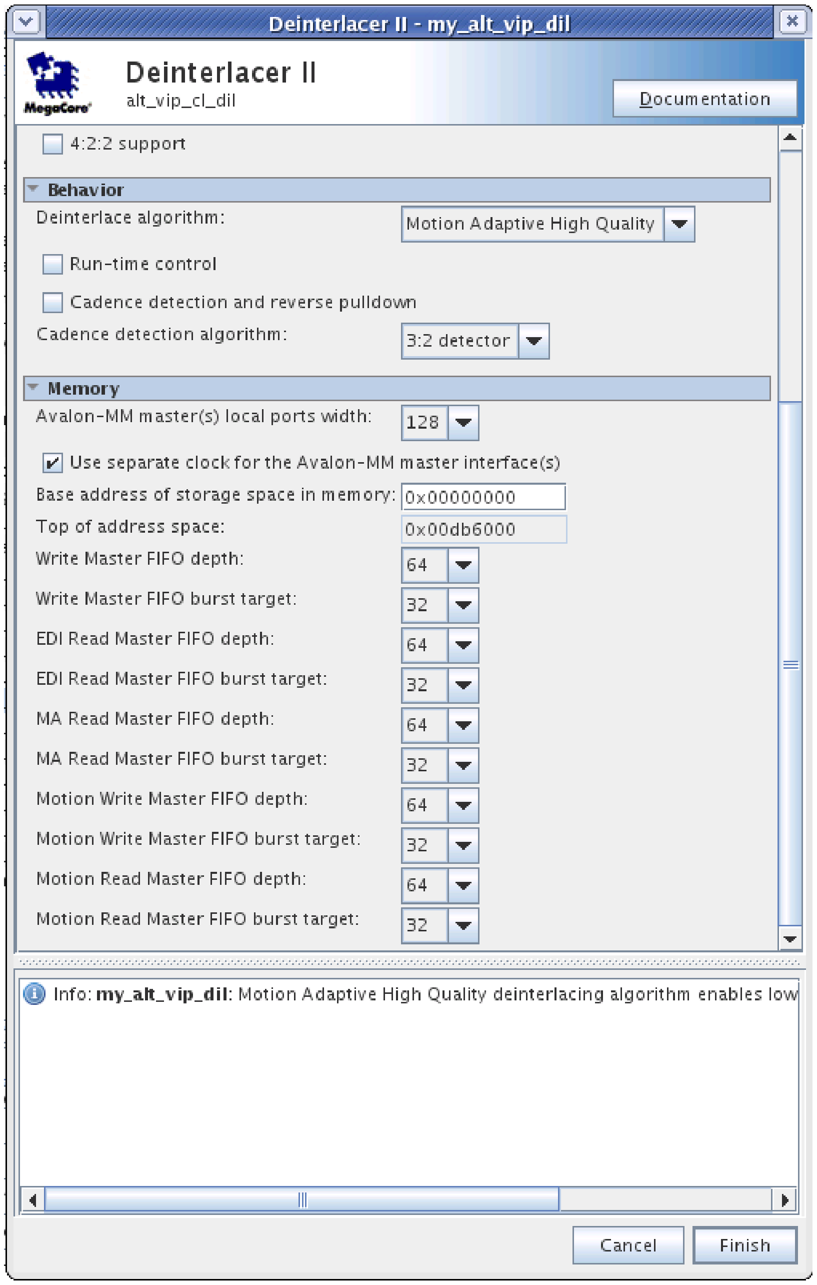
Deinterlace algorithm: Motion Adaptive High Quality
☐ Run-time control
☐ Cadence detection and reverse pulldown
Cadence detection algorithm: 3:2 detector

Avalon-MM master(s) local ports width: 128
☑ Use separate clock for the Avalon-MM master interface(s)
Base address of storage space in memory: 0x00000000
Top of address space: 0x00db6000

Info: **my_alt_vip_dil**: Motion Adaptive High Quality deinterlacing algorithm enables low

Figure 15 shows the additional parameters in the MegaWizard interface for the Clocked Video Input.

Figure 15. Additional Parameter Settings for the Clocked Video Input



Deinterlacer II - my_alt_vip_dil

Deinterlacer II
alt_vip_cl_dil

☐ 4:2:2 support

Behavior

Deinterlace algorithm: Motion Adaptive High Quality

☐ Run-time control

☐ Cadence detection and reverse pulldown

Cadence detection algorithm: 3:2 detector

Memory

Avalon-MM master(s) local ports width: 128

☑ Use separate clock for the Avalon-MM master interface(s)

Base address of storage space in memory:	0x00000000
Top of address space:	0x00db6000
Write Master FIFO depth:	64
Write Master FIFO burst target:	32
EDI Read Master FIFO depth:	64
EDI Read Master FIFO burst target:	32
MA Read Master FIFO depth:	64
MA Read Master FIFO burst target:	32
Motion Write Master FIFO depth:	64
Motion Write Master FIFO burst target:	32
Motion Read Master FIFO depth:	64
Motion Read Master FIFO burst target:	32

Info: **my_alt_vip_dil**: Motion Adaptive High Quality deinterlacing algorithm enables low

In this design example, the Deinterlacer II applies a motion adaptive algorithm. Because the algorithm uses information from previous fields to compute the progressive frame pixel values as a function of motion between fields, previous fields must be stored in external memory.

In this mode, the Deinterlacer II has five Avalon-MM master ports (two to write, and three to read data), which are connected to an external memory with enough space to store multiple frames of video data and one frame of motion data.

The base address of the Deinterlacer II frame buffers is 0x0 and represents the offset from the DDR2 Memory Controller Avalon-MM base address where the base of the frame buffer memory is to be located. The amount of free memory required at this location is displayed in the parameterization page.

The Deinterlacer II also allows you to specify a burst length target for writing and reading data to external memory. This combined with configurability of the FIFO depths between the Avalon-MM masters and the data path, provides you with the ability to buffer data in external memory without stalling the data path.

8. The progressive data output from the Deinterlacer II is clipped by a second Clipper MegaCore function. Double-click **my_alt_vip_clip_1** in Qsys to display the MegaWizard interface **Parameter Settings** page for the Clipper (Figure 16).

Figure 16. Parameter Settings for the Second Clipper

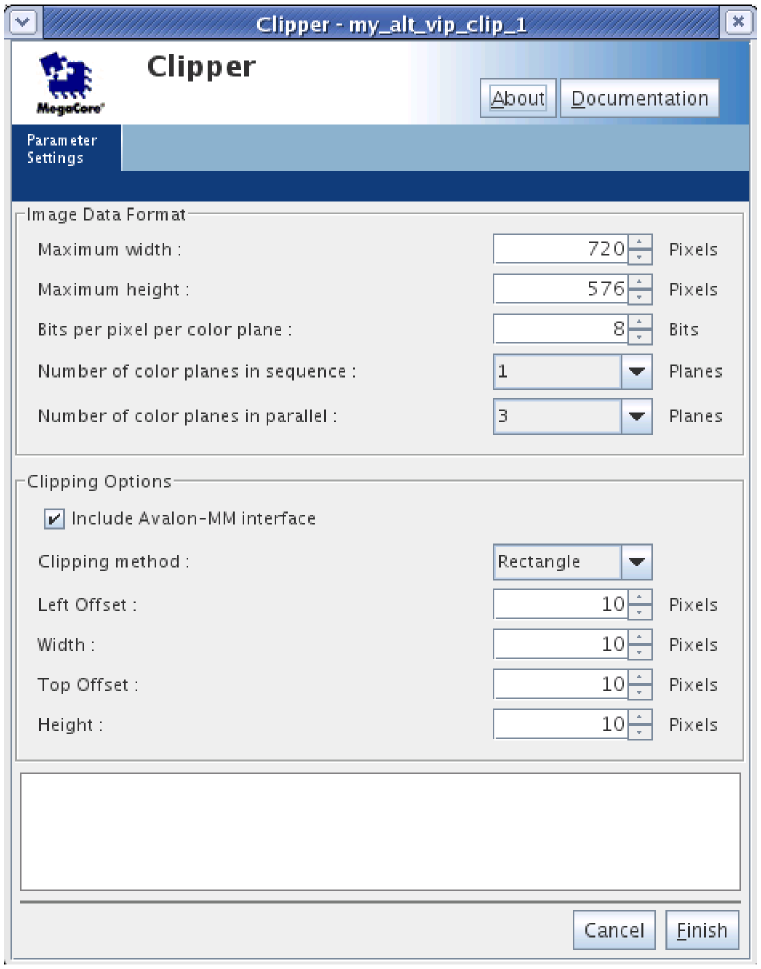


Image Data Format

Maximum width :	720	Pixels
Maximum height :	576	Pixels
Bits per pixel per color plane :	8	Bits
Number of color planes in sequence :	1	Planes
Number of color planes in parallel :	3	Planes

Clipping Options

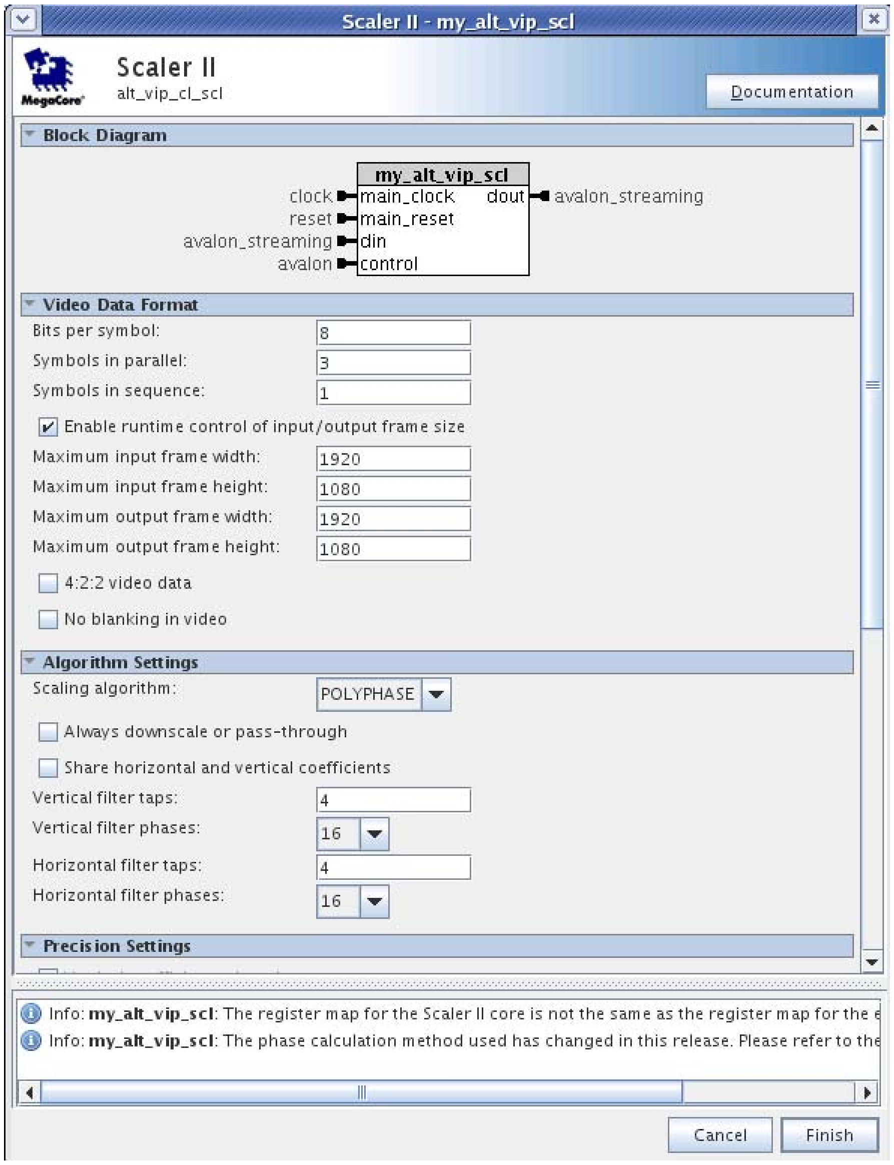
[✓] Include Avalon-MM interface

Clipping method :	Rectangle	
Left Offset :	10	Pixels
Width :	10	Pixels
Top Offset :	10	Pixels
Height :	10	Pixels

The clipping region is specified at run-time using an Avalon-MM slave interface.

9. The clipped video output from the second Clipper is upscaled by the Scaler II MegaCore function. Double-click **my_alt_vip_scl** in Qsys to display the MegaWizard interface **Resolution Parameter Settings** page for the Scaler II MegaCore function (Figure 17).

Figure 17. Resolution Parameter Settings for the Scaler II



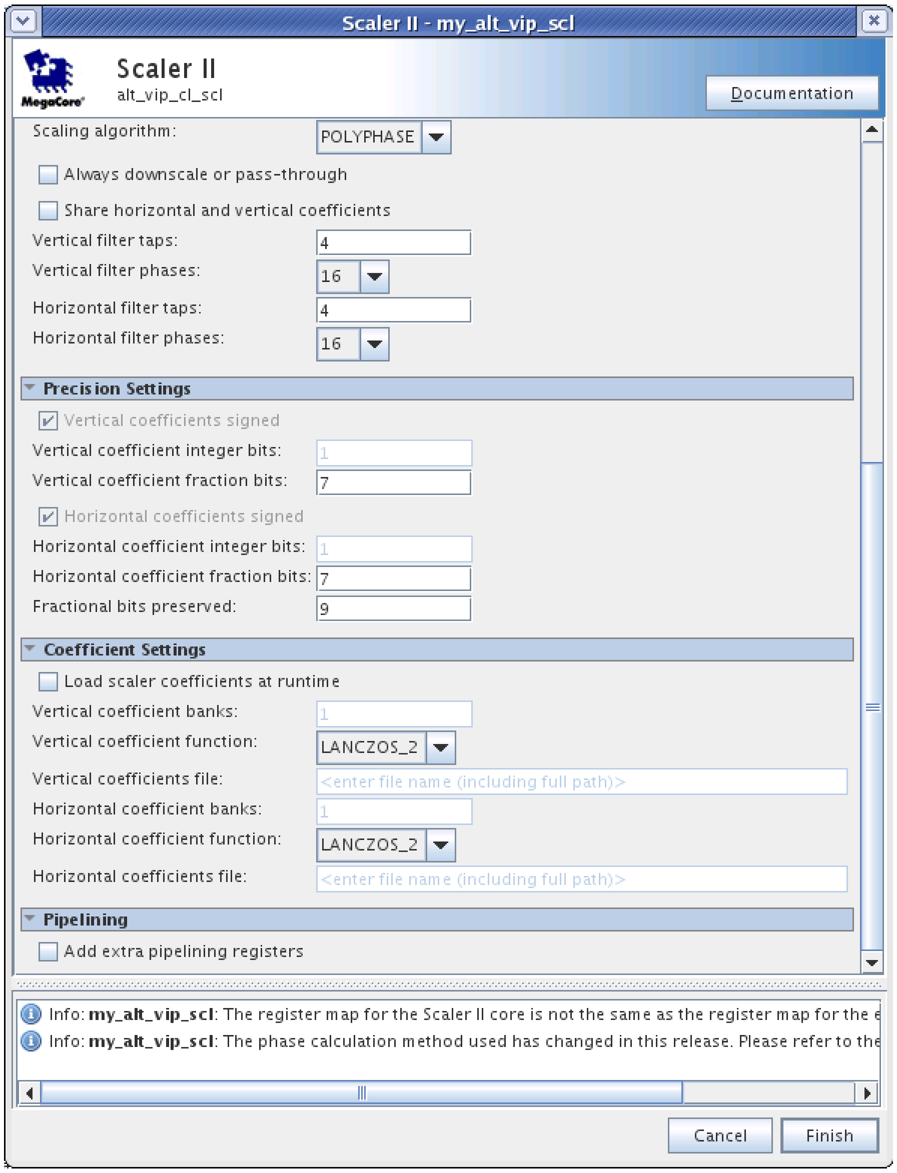
Bits per symbol: 8
Symbols in parallel: 3
Symbols in sequence: 1
[✓] Enable runtime control of input/output frame size
Maximum input frame width: 1920
Maximum input frame height: 1080
Maximum output frame width: 1920
Maximum output frame height: 1080
[ ] 4:2:2 video data
[ ] No blanking in video

Scaling algorithm: POLYPHASE
[ ] Always downscale or pass-through
[ ] Share horizontal and vertical coefficients
Vertical filter taps: 4
Vertical filter phases: 16
Horizontal filter taps: 4
Horizontal filter phases: 16

Info: **my_alt_vip_scl**: The register map for the Scaler II core is not the same as the register map for the
Info: **my_alt_vip_scl**: The phase calculation method used has changed in this release. Please refer to the

Figure 18 shows the additional parameters in the MegaWizard interface for the Scalar II.

Figure 18. Resolution Parameter Settings for the Scaler II, Continued



Scaling algorithm: POLYPHASE
☐ Always downscale or pass-through
☐ Share horizontal and vertical coefficients
Vertical filter taps: 4
Vertical filter phases: 16
Horizontal filter taps: 4
Horizontal filter phases: 16

☑ Vertical coefficients signed
Vertical coefficient integer bits: 1
Vertical coefficient fraction bits: 7
☑ Horizontal coefficients signed
Horizontal coefficient integer bits: 1
Horizontal coefficient fraction bits: 7
Fractional bits preserved: 9

☐ Load scaler coefficients at runtime
Vertical coefficient banks: 1
Vertical coefficient function: LANCZOS_2
Vertical coefficients file: <enter file name (including full path)>
Horizontal coefficient banks: 1
Horizontal coefficient function: LANCZOS_2
Horizontal coefficients file: <enter file name (including full path)>

☐ Add extra pipelining registers

Info: **my_alt_vip_scl**: The register map for the Scaler II core is not the same as the register map for the e
Info: **my_alt_vip_scl**: The phase calculation method used has changed in this release. Please refer to the

The Scaler II is configured to allow run-time control of the scaler output width and height using an Avalon-MM slave interface up to a maximum of 1920×1080 pixels.

The Scaler II uses the Polyphase scaling algorithm (configured for four vertical and four horizontal taps) with 16 vertical and horizontal phases.

10. The scaled video stream is then buffered in external memory using the Frame Buffer MegaCore function. Double-click **my_alt_vip_vfb** in Qsys to display the MegaWizard interface **Parameter Settings** page for the Frame Buffer (Figure 19).

    The maximum resolution supported by this Frame Buffer instance is 1920×1080 pixels. The Frame Buffer streams data from the Avalon-ST sink interface into a dual clock FIFO.

Figure 19. Parameter Settings for the Frame Buffer

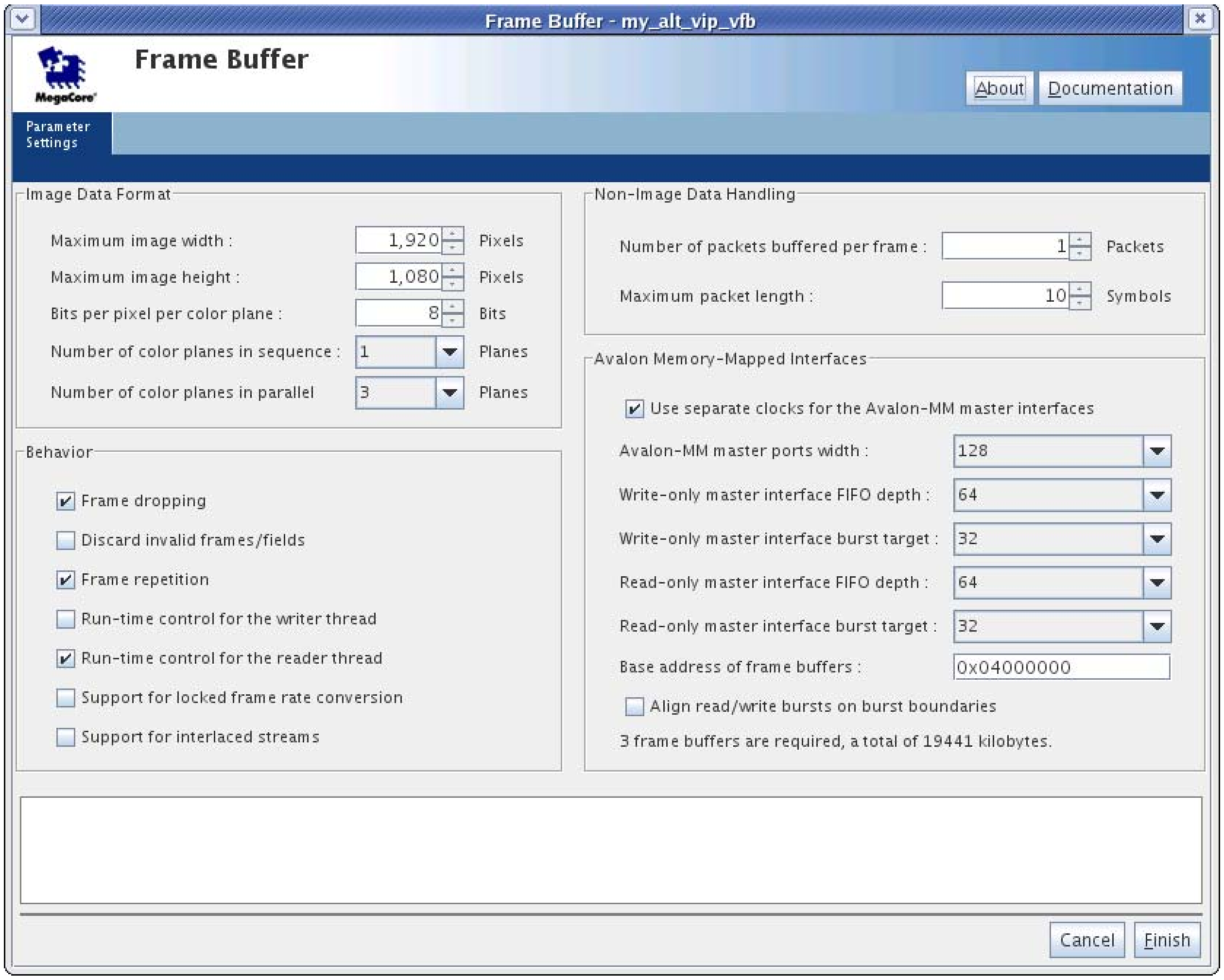


Image Data Format

Maximum image width :	1,920	Pixels
Maximum image height :	1,080	Pixels
Bits per pixel per color plane :	8	Bits
Number of color planes in sequence :	1	Planes
Number of color planes in parallel	3	Planes

Behavior

- ✓ Frame dropping
- ☐ Discard invalid frames/fields
- ✓ Frame repetition
- ☐ Run-time control for the writer thread
- ✓ Run-time control for the reader thread
- ☐ Support for locked frame rate conversion
- ☐ Support for interlaced streams

Non-Image Data Handling

Number of packets buffered per frame :	1	Packets
Maximum packet length :	10	Symbols

Avalon Memory-Mapped Interfaces

✓ Use separate clocks for the Avalon-MM master interfaces

Avalon-MM master ports width :	128
Write-only master interface FIFO depth :	64
Write-only master interface burst target :	32
Read-only master interface FIFO depth :	64
Read-only master interface burst target :	32
Base address of frame buffers :	0x04000000

☐ Align read/write bursts on burst boundaries

3 frame buffers are required, a total of 19441 kilobytes.

In this example, the Frame Buffer is set to support rate conversion by dropping and repeating frames.

When the number of data items in the FIFO is above the burst target, the Avalon-MM bursting write master drives a write request to external memory, with a burst length typically equal to the burst target. Similarly, the bursting read master drives a read request to read data with a burst length typically of length equal to the burst target.

There are many challenges associated with building video systems. One common design requirement is to achieve the mandatory level of external memory bandwidth when multiple components are storing video frames in a shared DDR external memory. This requirement often necessitates a flexible system, where the data processing rate can be decoupled from external memory clock rate constraints.

Both the Frame Buffer and Deinterlacer functions provide built-in clock crossing capability using dual clock FIFOs to separate the data processing clock domain from the external memory controller clock domain.

Data is input via the Avalon-ST sink interface as 24-bit wide data (three colors in parallel). 48 bits (two pixels) of data is written to/read from external memory via an external memory port width of 64 bits. The base address of the frame buffer is 0x4000000 and represents the offset in external DDR2 memory where the base of the frame buffer memory is located. The amount of free memory required at this location is displayed in the parameterization page.

In addition, the run-time control of the reader thread is enabled, which provides an Avalon-MM slave port. In this system, the Nios II processor data master is connected to the Avalon-MM slave. This connection allows the Nios II processor to control when the Frame Buffer starts outputting data.

11. The buffered video stream is then mixed (picture-in-picture) with a test pattern image. Double-click **my_alt_vip_mix** in Qsys to display the MegaWizard interface **Parameter Settings** page for the Alpha Blending Mixer (Figure 20).

Figure 20. Parameter Settings for the Alpha Blending Mixer

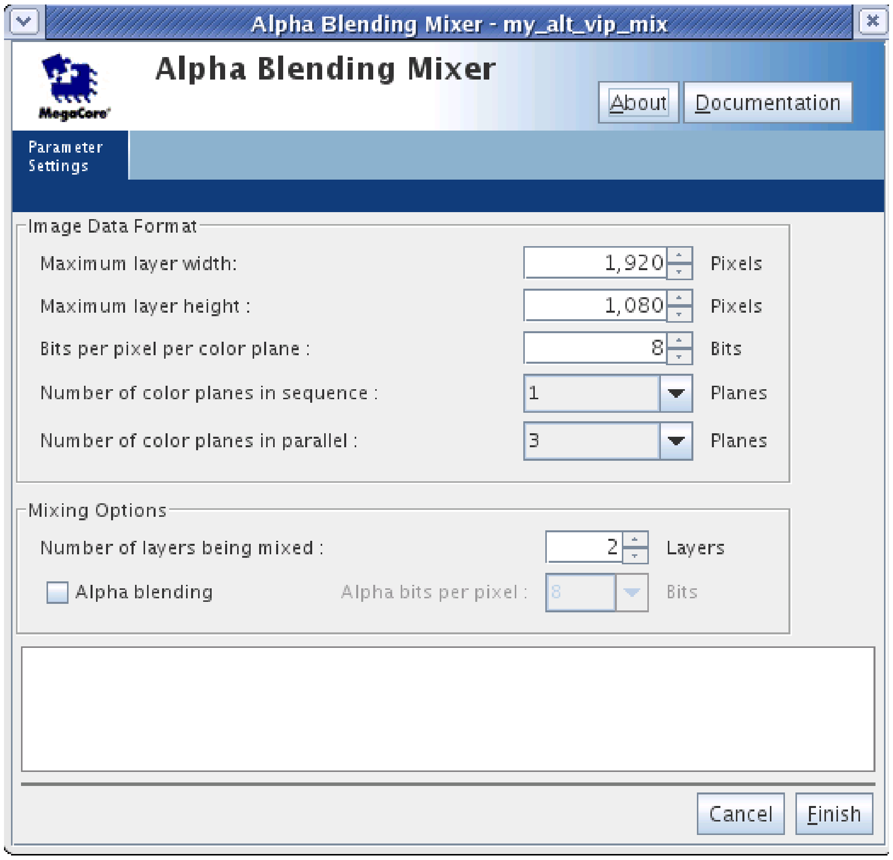


Image Data Format

Maximum layer width:	1,920	Pixels
Maximum layer height :	1,080	Pixels
Bits per pixel per color plane :	8	Bits
Number of color planes in sequence :	1	Planes
Number of color planes in parallel :	3	Planes

Mixing Options

Number of layers being mixed :	2	Layers
☐ Alpha blending — Alpha bits per pixel :	8	Bits

The test pattern is input as the background layer, with a resolution of 1920×1080 pixels. The Alpha Blending Mixer is configured to mix 2 layers without alpha blending. This provides a picture-in-picture effect.

The location of the foreground layer, relative to the background layer, is specified at run-time through the Avalon-MM Slave interface. In this example, the Nios II processor writes control data to the mixer and other functions in the data path.

12. The test pattern is generated by the Test Pattern Generator MegaCore function. Double-click **my_alt_vip_tpg** in Qsys to display the MegaWizard interface **Parameter Settings** page for the Test Pattern Generator (Figure 21).

Figure 21. Parameter Settings for the Test Pattern Generator

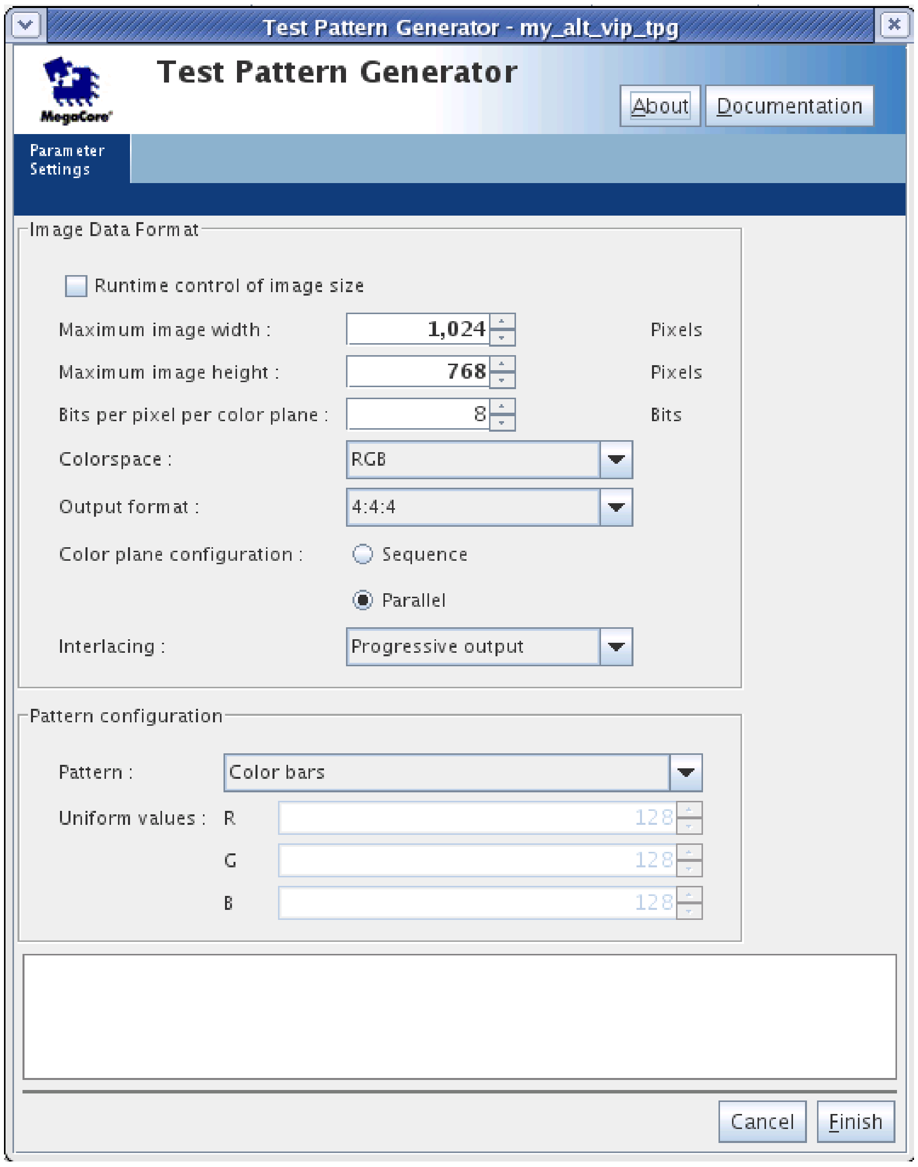


Image Data Format

Runtime control of image size

Maximum image width :	1,024	Pixels
Maximum image height :	768	Pixels
Bits per pixel per color plane :	8	Bits
Colorspace :	RGB	
Output format :	4:4:4	
Color plane configuration :	Sequence [✓ Parallel]	
Interlacing :	Progressive output	

Pattern configuration

Pattern :	Color bars	
Uniform values :	R	128
	G	128
	B	128

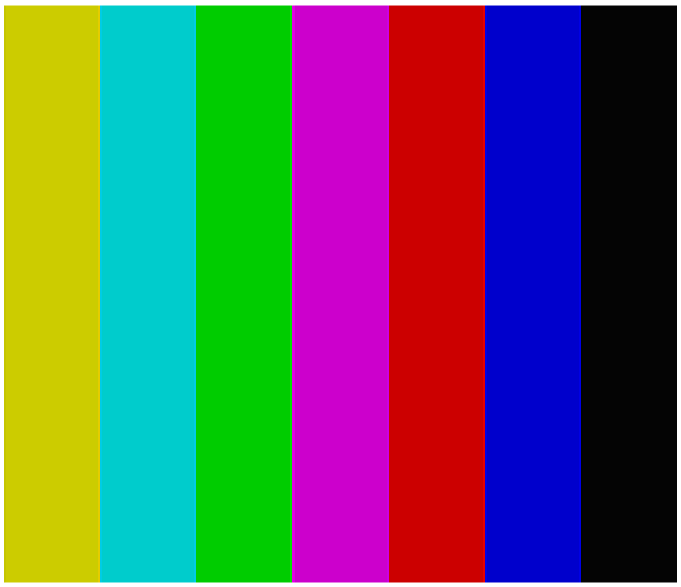
The Test Pattern Generator is a stallable video source that supplies data when requested. It enables you to develop and validate your data path and video outputs, before building and integrating the video sources into the system.

The Test Pattern Generator can produce a standard test pattern with the following options:

- RGB or YCbCr color space
- Full sampled data (4:4:4) or sub-sampled data (4:2:2 or 4:2:0)
- Progressive or interlaced
- Colors in sequence or colors in parallel

The generated test pattern is shown in Figure 22.

Figure 22. Generated Test Pattern



13. The mixed image is streamed to the Clocked Video Output MegaCore function, which forms the video stream output boundary to the Qsys system. Double-click **my_alt_vip_itc** in Qsys to display the MegaWizard interface for Clocked Video Output, shown in Figure 23.

    The Clocked Video Output MegaCore function converts from the Avalon-ST Video protocol to clocked video formats (such as BT656 and DVI). It formats Avalon-ST Video into clocked video by inserting horizontal and vertical blanking and generating horizontal and vertical sync information using the Avalon-ST Video control and active picture packets.

    In this example, the Clocked Video Output function is configured to accept video data with three color planes in parallel, namely, RGB. The function also outputs clocked video data with the sync signals on separate wires, suitable to transmit data over a DVI interface.

Figure 23. Parameter Settings for the Clocked Video Output

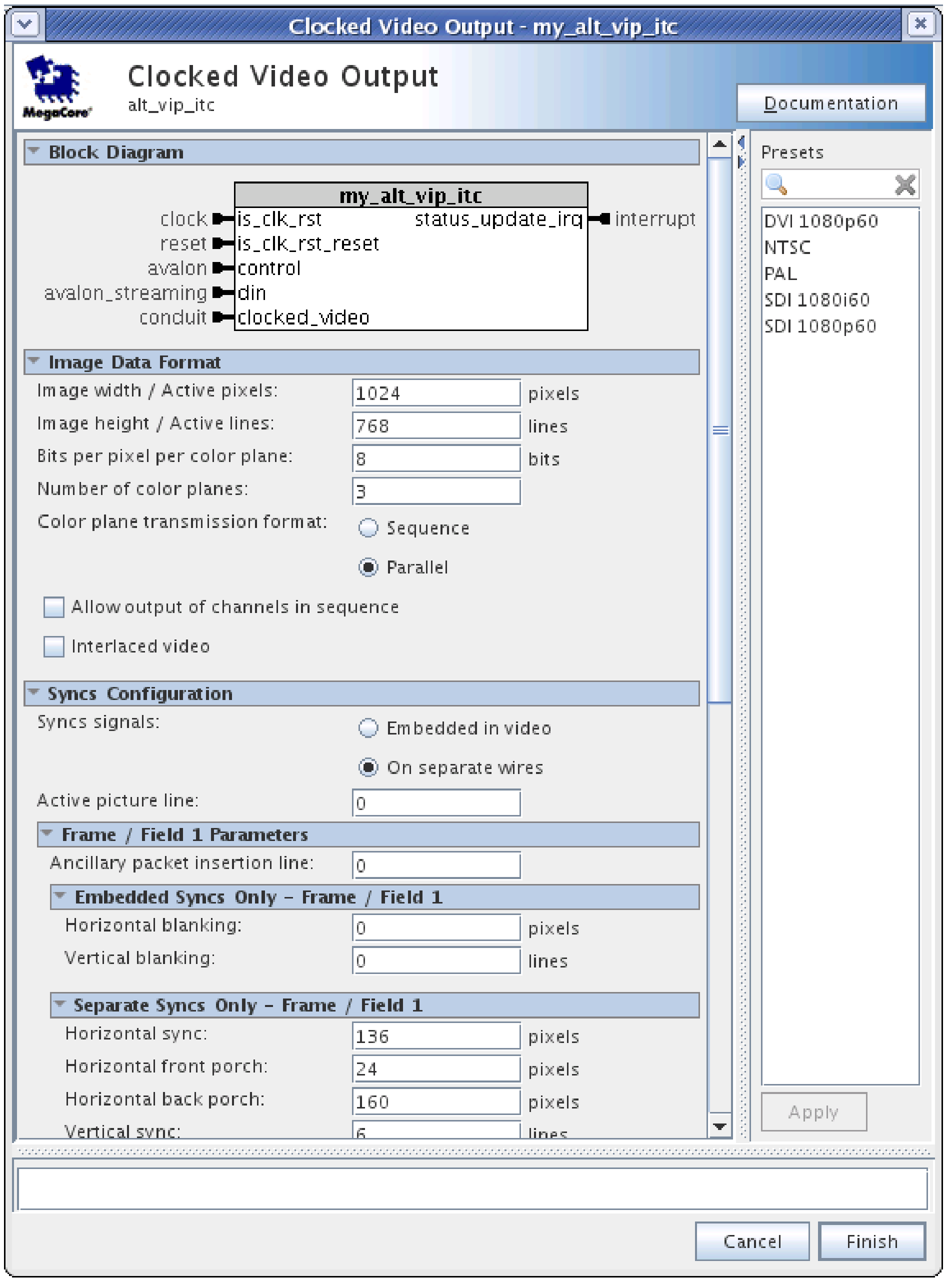


Image Data Format

Image width / Active pixels: 1024 pixels

Image height / Active lines: 768 lines

Bits per pixel per color plane: 8 bits

Number of color planes: 3

Color plane transmission format: Sequence [✓ Parallel]

☐ Allow output of channels in sequence

☐ Interlaced video

Syncs Configuration

Syncs signals: Embedded in video [✓ On separate wires]

Active picture line: 0

Frame / Field 1 Parameters

Ancillary packet insertion line: 0

Embedded Syncs Only - Frame / Field 1

Horizontal blanking: 0 pixels

Vertical blanking: 0 lines

Separate Syncs Only - Frame / Field 1

Horizontal sync: 136 pixels

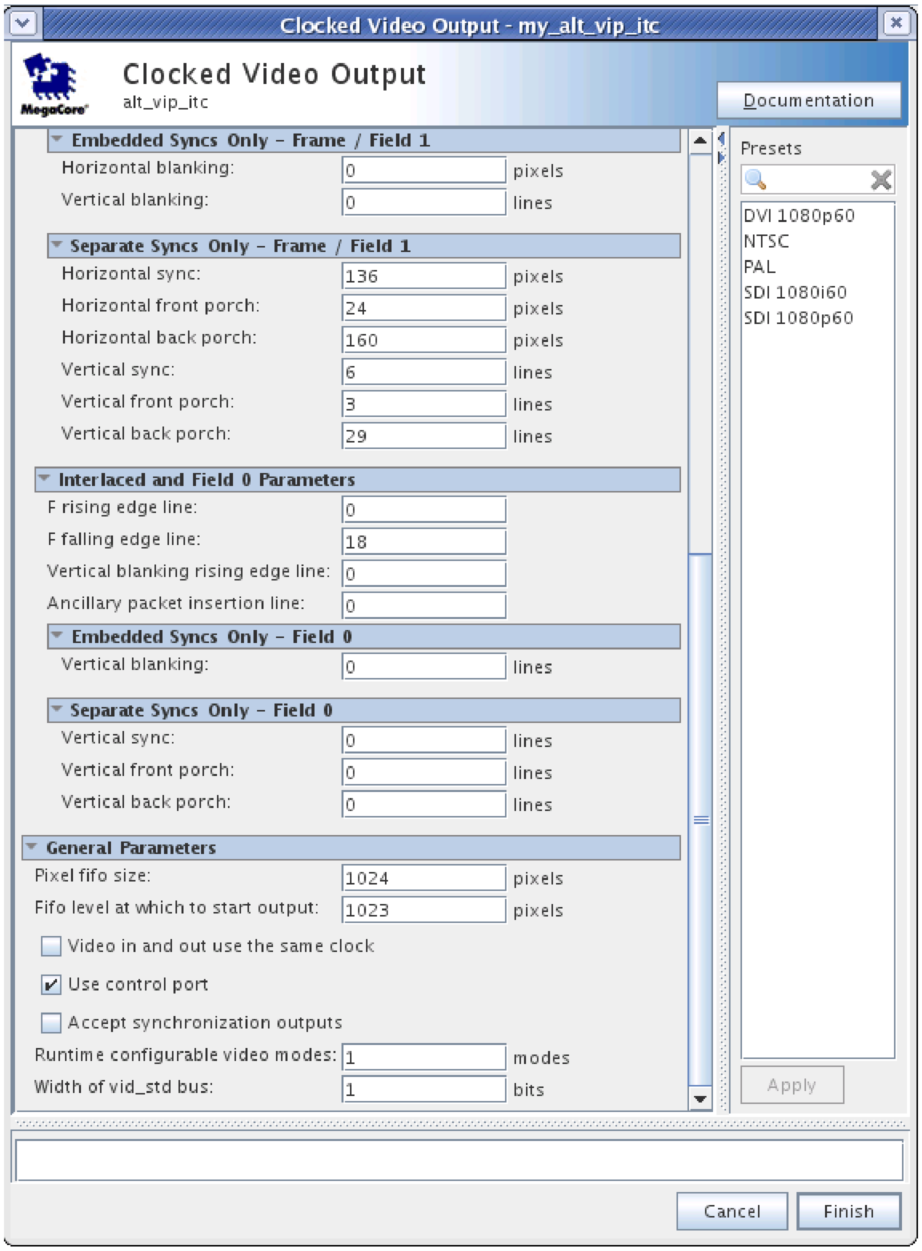
Horizontal front porch: 24 pixels

Horizontal back porch: 160 pixels

Vertical sync: 6 lines

Figure 24 shows the additional parameters in the MegaWizard interface for Clocked Video Output.

Figure 24. Parameter Settings for the Clocked Video Output, Continued



Clocked Video Output
alt_vip_itc

Embedded Syncs Only – Frame / Field 1

Horizontal blanking:	0	pixels
Vertical blanking:	0	lines

Separate Syncs Only – Frame / Field 1

Horizontal sync:	136	pixels
Horizontal front porch:	24	pixels
Horizontal back porch:	160	pixels
Vertical sync:	6	lines
Vertical front porch:	3	lines
Vertical back porch:	29	lines

Interlaced and Field 0 Parameters

F rising edge line:	0
F falling edge line:	18
Vertical blanking rising edge line:	0
Ancillary packet insertion line:	0

Embedded Syncs Only – Field 0

Vertical blanking:	0	lines

Separate Syncs Only – Field 0

Vertical sync:	0	lines
Vertical front porch:	0	lines
Vertical back porch:	0	lines

General Parameters

Pixel fifo size:	1024	pixels
Fifo level at which to start output:	1023	pixels

- [ ] Video in and out use the same clock
- [✓] Use control port
- [ ] Accept synchronization outputs

Runtime configurable video modes:	1	modes
Width of vid_std bus:	1	bits

Presets:
DVI 1080p60
NTSC
PAL
SDI 1080i60
SDI 1080p60

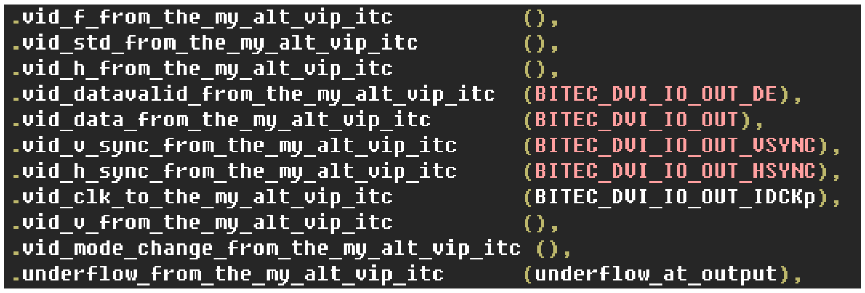
The **Video in and out use the same clock** option is turned off, which means that the Clocked Video Input block is configured to use separate clocks for the clocked video side and the flow controlled domain. The output pixel clock is set to 65MHz and the system clock is 100MHz.

The **Use control port** option is turned on, which provides an Avalon-MM slave port. In this system, the Nios II processor data master is connected to the Avalon-MM slave, which gives the Nios II processor control when the Clocked Video Output starts accepting data.

The register map also contains many status registers that provide feedback on the format of video entering the Clocked Video Output (resolution, interlaced or progressive). A status interrupt can also be used, but is not demonstrated in this example.

The clocked video signals are exported to the top-level system for connection to the output pins for the Bitec HSMC DVI input/output daughtercard. These exported signals are shown in the `System` instantiation in the file **vip_top.v** as shown in Figure 25.

Figure 25. Signal Exported from the Clocked Video Output



```
.vid_f_from_the_my_alt_vip_itc          (),
.vid_std_from_the_my_alt_vip_itc        (),
.vid_h_from_the_my_alt_vip_itc          (),
.vid_datavalid_from_the_my_alt_vip_itc  (BITEC_DVI_IO_OUT_DE),
.vid_data_from_the_my_alt_vip_itc       (BITEC_DVI_IO_OUT),
.vid_v_sync_from_the_my_alt_vip_itc     (BITEC_DVI_IO_OUT_VSYNC),
.vid_h_sync_from_the_my_alt_vip_itc     (BITEC_DVI_IO_OUT_HSYNC),
.vid_clk_to_the_my_alt_vip_itc          (BITEC_DVI_IO_OUT_IDCKp),
.vid_v_from_the_my_alt_vip_itc          (),
.vid_mode_change_from_the_my_alt_vip_itc (),
.underflow_from_the_my_alt_vip_itc      (underflow_at_output),
```

I2C Bus Masters (for Configuration of Video Input/Output Chips)

In this design example, two OpenCore I2C bus masters are instantiated to enable configuration of both the TVPS154 and TFP410 video interface chips on the HSMC Quad Video and HSMC DVI input/output daughtercards, respectively.

The I2C bus master components provide a convenient interface to configure the video interface chips. An Avalon-MM slave interface receives read and write commands, which are translated into I2C bus transactions.

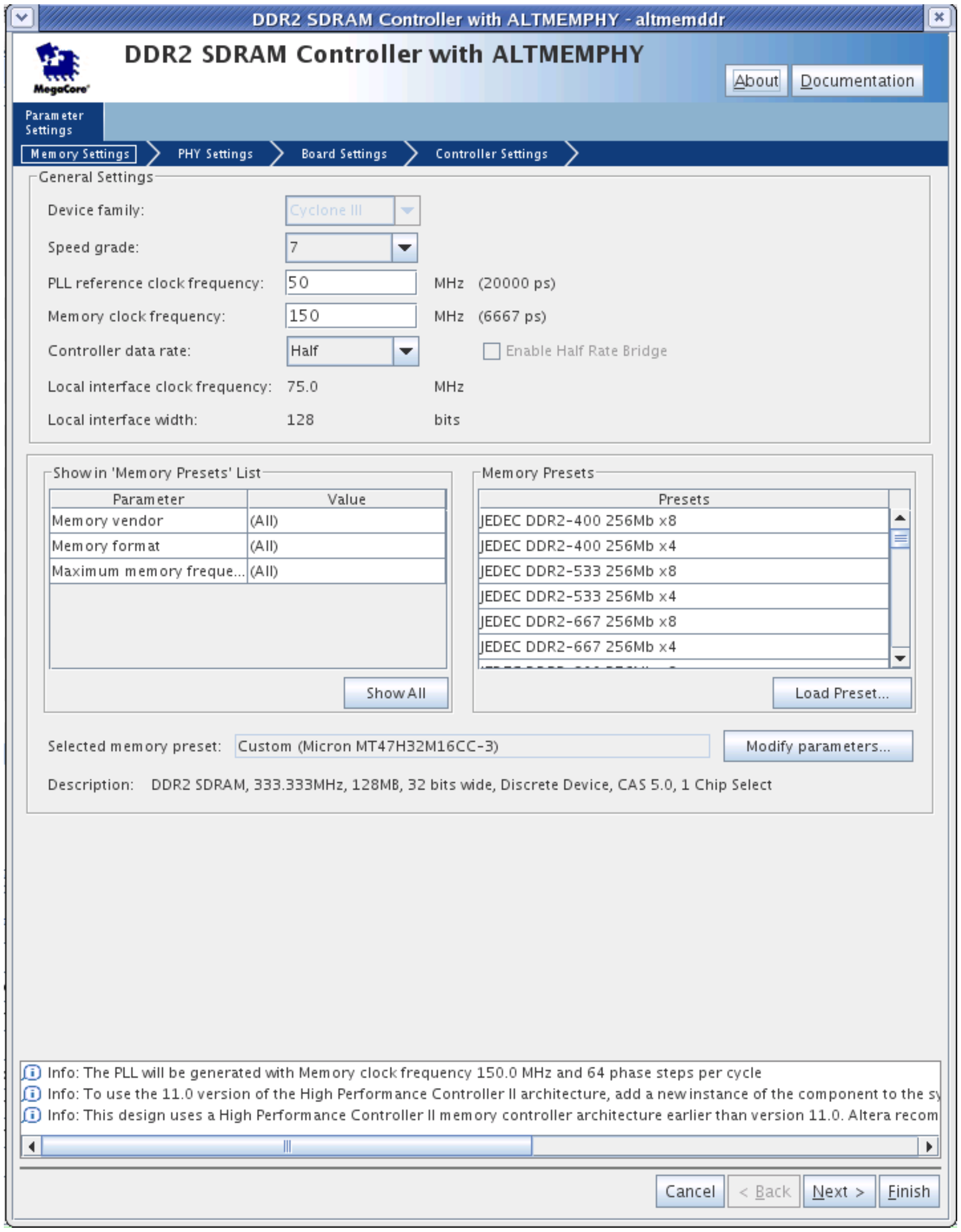
The Nios II processor controls the configuration of the video interface chips by writing and reading data to and from the I2C bus master component. The I2C bus masters are not parameterizable through a user interface in Qsys. The HDL source code is provided in the local project directory (**ip/***).

DDR2 External Memory Controller (for Video Buffering)

The Altera DDR2 SDRAM High Performance Controller II controls the data flow to and from the external DDR2 memory. The controller is parameterized in half-rate mode, operating at 150 MHz, with a local data interface width of 128 bits. This configuration provides a theoretical maximum available bandwidth of 75×128 = 9.6 GBits/sec. The 100MHz system clock is used as the PLL reference clock.

Double-click **altmemddr** to display the Parameter Settings page for the DDR2 SDRAM High Performance Controller II (Figure 26).

Figure 26. Parameter Settings for the DDR2 SDRAM High Performance Controller II



DDR2 SDRAM Controller with ALTMEMPHY

General Settings

Device family:	Cyclone III	
Speed grade:	7	
PLL reference clock frequency:	50	MHz (20000 ps)
Memory clock frequency:	150	MHz (6667 ps)
Controller data rate:	Half	Enable Half Rate Bridge
Local interface clock frequency:	75.0	MHz
Local interface width:	128	bits

Show in 'Memory Presets' List

Parameter	Value
Memory vendor	(All)
Memory format	(All)
Maximum memory freque...	(All)

Memory Presets

Presets
JEDEC DDR2-400 256Mb x8
JEDEC DDR2-400 256Mb x4
JEDEC DDR2-533 256Mb x8
JEDEC DDR2-533 256Mb x4
JEDEC DDR2-667 256Mb x8
JEDEC DDR2-667 256Mb x4

Selected memory preset: Custom (Micron MT47H32M16CC-3)

Description: DDR2 SDRAM, 333.333MHz, 128MB, 32 bits wide, Discrete Device, CAS 5.0, 1 Chip Select

```
Info: The PLL will be generated with Memory clock frequency 150.0 MHz and 64 phase steps per cycle
Info: To use the 11.0 version of the High Performance Controller II architecture, add a new instance of the component to the sy
Info: This design uses a High Performance Controller II memory controller architecture earlier than version 11.0. Altera recom
```

Nios II Processor and On-Chip Memory for Program Code

The Nios II processor initializes and configures multiple video system components, including the I2C external bus masters and the Video and Image Processing Suite MegaCore functions.

The Nios II Avalon-MM data master is connected to each Avalon-MM slave port it needs access to. The bus architecture is automatically constructed by Qsys. The system is designed such that the Nios II program code is stored in on-chip memory, with 100 Kbytes of available space.

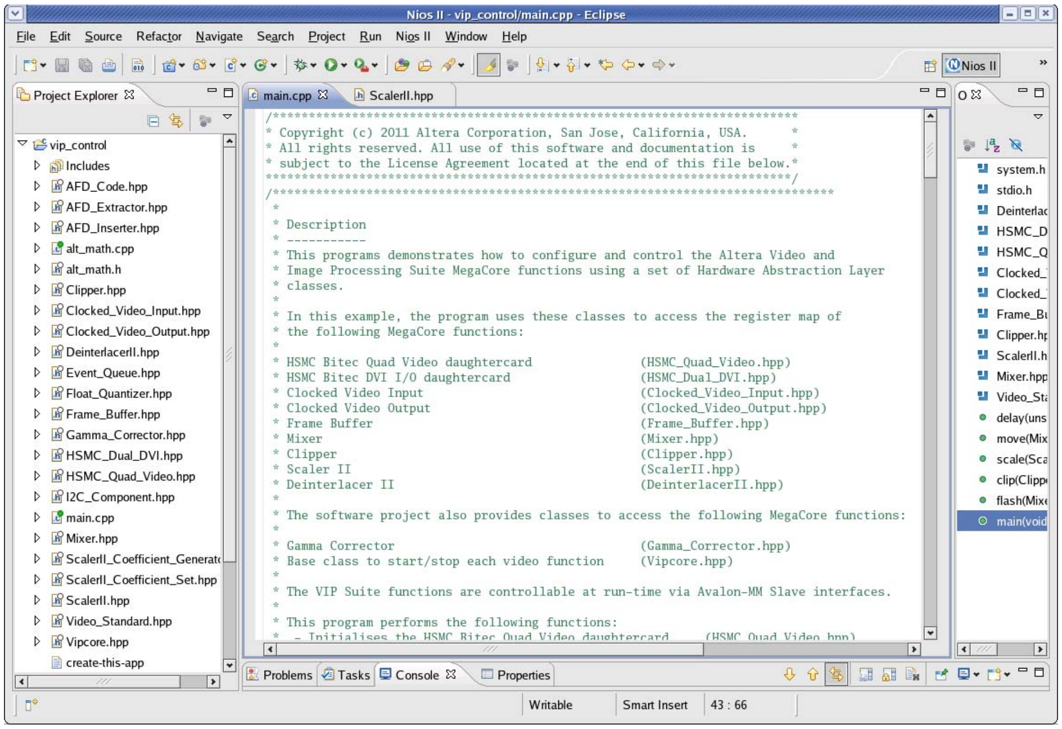
Nios II Software

Nios II software is provided to perform the following functions:

- Configuration of the Bitec Quad Video input daughtercard video chips (TVP5154 and Chrontel)
- Configuration of the Bitec DVI input/output daughtercard video chips (TFP410)
- Initialize and start the Video and Image Processing Suite MegaCore functions via the Avalon-MM slave control interfaces
- Control the Clipper function to change the height and width of the clipped video region
- Control the Scaler function output resolution
- Control the mixer function to change the location of the video stream relative to the test pattern background layer

The **vip_control** application project contains a description of the application in the **main.cpp** file as shown in Figure 27.

Figure 27. main.cpp in Nios II SBT for Eclipse



```
/*****************************************************************************
* Copyright (c) 2011 Altera Corporation, San Jose, California, USA.        *
* All rights reserved. All use of this software and documentation is       *
* subject to the License Agreement located at the end of this file below.*
****************************************************************************/
/******************************************************************************
 *
 * Description
 * -----------
 * This programs demonstrates how to configure and control the Altera Video and
 * Image Processing Suite MegaCore functions using a set of Hardware Abstraction Layer
 * classes.
 *
 * In this example, the program uses these classes to access the register map of
 * the following MegaCore functions:
 *
 * HSMC Bitec Quad Video daughtercard               (HSMC_Quad_Video.hpp)
 * HSMC Bitec DVI I/O daughtercard                  (HSMC_Dual_DVI.hpp)
 * Clocked Video Input                              (Clocked_Video_Input.hpp)
 * Clocked Video Output                             (Clocked_Video_Output.hpp)
 * Frame Buffer                                     (Frame_Buffer.hpp)
 * Mixer                                            (Mixer.hpp)
 * Clipper                                          (Clipper.hpp)
 * Scaler II                                        (ScalerII.hpp)
 * Deinterlacer II                                  (DeinterlacerII.hpp)
 *
 * The software project also provides classes to access the following MegaCore functions:
 *
 * Gamma Corrector                                  (Gamma_Corrector.hpp)
 * Base class to start/stop each video function     (Vipcore.hpp)
 *
 * The VIP Suite functions are controllable at run-time via Avalon-MM Slave interfaces.
 *
 * This program performs the following functions:
 *  - Initialises the HSMC Bitec Quad Video daughtercard     (HSMC_Quad_Video.hpp)
```

C++ classes are included in the **vip_control** project that provide a software API between the Nios II control code and the Video and Image Processing Suite MegaCore functions. The classes provide many member functions to accelerate object-oriented code development and increase visibility of the data flow. For example, the clocked video input class (**Clocked_Video_Input.hpp**) member functions can report the level in a FIFO in the video system, or the member functions in the frame buffer class (**Frame_Buffer.hpp**) can report the number of frames that are dropped or repeated by the Frame Buffer in the data path.

All Video and Image Processing Suite MegaCore function classes are derived from the base class **Vipcore.hpp**, which contains methods common to all functions such as starting and stopping the video function processing at a frame boundary.

The `main` function in **main.cpp** provides examples of how to use the C++ class member functions to configure and control the data path functions, as shown in Example 1.

Example 1. Configuring and Controlling Data Path Functions in main.cpp (Sheet 1 of 3)

```
int main(void)
{  /* Initialise the HSMC Bitec DVI I/O daughtercard */
   HSMC_Dual_DVI dual_dvi_board(DVI_OUTPUT_I2C_MASTER_BASE);
   dual_dvi_board.enable_output();

   /* Initialise the HSMC Bitec Quad Video daughtercard */
   HSMC_Quad_Video quad_analog_in_board(QUAD_VIDEO_I2C_MASTER_BASE);
   quad_analog_in_board.init();

   /* Start the Frame Buffer MegaCore function */
   Frame_Buffer_Reader the_frame_buffer(MY_ALT_VIP_VFB_BASE);
   the_frame_buffer.start();

   /* Initialise and start the Alpha Blending Mixer MegaCore function */
   Mixer the_mixer(MY_ALT_VIP_MIX_BASE);
   the_mixer.set_layer_position(1,0,0);
   the_mixer.set_layer_enabled(1, true);
   the_mixer.start();

   /* Start the Scaler II MegaCore function */
   Scaler_II<4,4> the_scaler(MY_ALT_VIP_SCL_BASE, 16, 16);
   the_scaler.set_output_size(720, 480);
   the_scaler.start();

   /* Continued on next page */
```

Example 1. Configuring and Controlling Data Path Functions in main.cpp (Sheet 2 of 3)

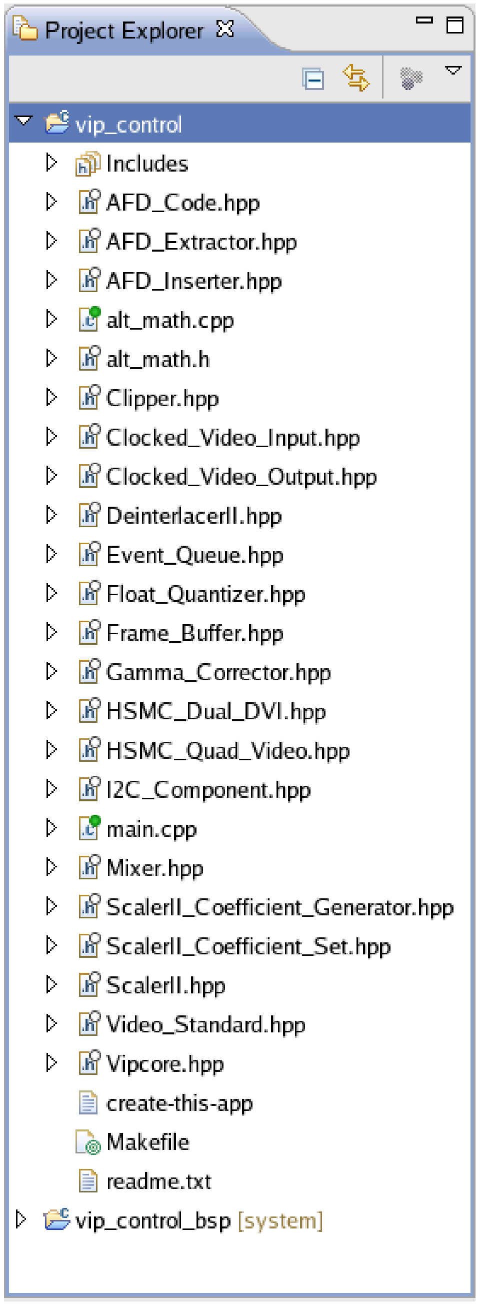
```
/* Continued from previous page */

/* Start the Clipper MegaCore function, clipping progressive video
* (after the deinterlacer function) */
Clipper_Rectangle_Type
                 the_progressive_clipper(MY_ALT_VIP_CLIP_1_BASE);
the_progressive_clipper.set_left_offset(0);
the_progressive_clipper.set_top_offset(0);
the_progressive_clipper.set_width(720);
the_progressive_clipper.set_height(480);
the_progressive_clipper.start();

/* Start the deinterlacer */
Deinterlacer II the_deinterlacer(MY_ALT_VIP_DIL_BASE);
the_deinterlacer.start();

/* Start the Clipper MegaCore function clipping the interlaced video.
* This clips the top 3 lines from the input NTSC or PAL video */
Clipper_Rectangle_Type the_interlaced_clipper(MY_ALT_VIP_CLIP_BASE);
the_interlaced_clipper.set_left_offset(0);
the_interlaced_clipper.set_top_offset(3);
the_interlaced_clipper.set_width(720);
the_interlaced_clipper.set_height(240);
the_interlaced_clipper.start();

/* Start the Clocked Video Input MegaCore function */
Clocked_Video_Input the_clocked_video_input(Color_Space::YCbCrSD,
8, MY_ALT_VIP_CTI_BASE, MY_ALT_VIP_CTI_IRQ);
while (the_clocked_video_input.is_input_stable() == false)
{}
the_clocked_video_input.clear sticky_overflow_bit();
the_clocked_video_input.start();

printf("F0 width x height= %d x
%d\n",the_clocked_video_input.get_f0_sample_count(),
the_clocked_video_input.get_f0_line_count());
printf("F1 width x height= %d x
%d\n",the_clocked_video_input.get_f1_sample_count(),
the_clocked_video_input.get_f1_line_count());

/* Continued on next page */
```

Example 1. Configuring and Controlling Data Path Functions in main.cpp (Sheet 3 of 3)

```
   /* Continued from previous page */


   /* Start the Clocked Video Output MegaCore function */

   Clocked_Video_Output<1>
   the_clocked_video_output(MY_ALT_VIP_ITC_BASE);

   /* NB. It is not necessary to check the clocked video output FIFO
   before starting it up : */

   the_clocked_video_output.clear_fifo_underflow();

   the_clocked_video_output.start();


   while (1) {

      clip(the_progressive_clipper);

      scale(the_scaler);

      move(the_mixer, 1);

      flash(the_mixer, 1);

   }

}
```

Review the **.hpp** files included in the application project to understand the control capability from the Nios II software environment (Figure 28).

Figure 28. .hpp Files in the Nios II SBT for Eclipse

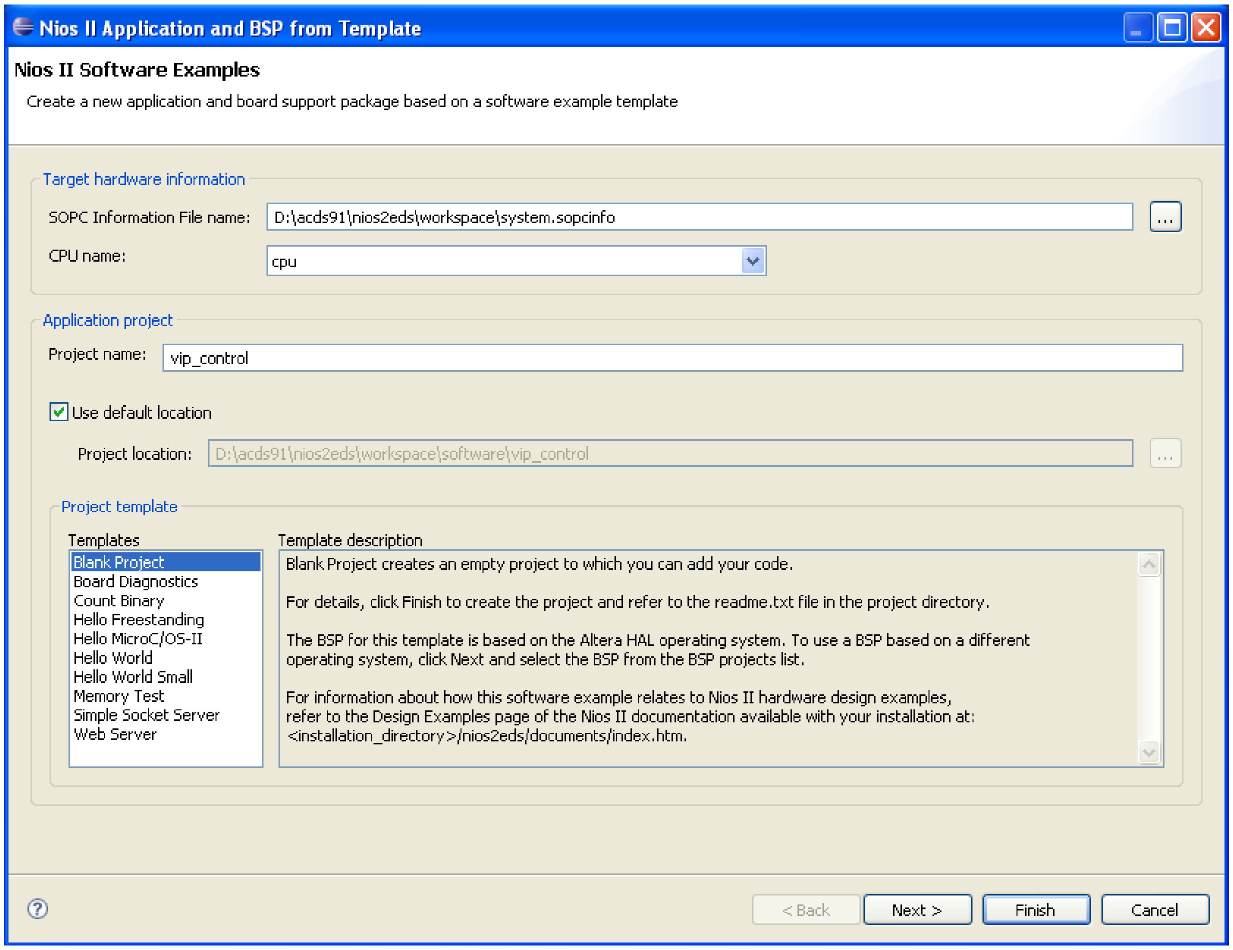


To examine and recompile the Nios II C++ control code and associated board support package (BSP) using the Nios II SBT for Eclipse, perform the following steps:

1. On the Windows Start menu, point to **All Programs**, point to **Altera**, point to **Nios II EDS <*version*>**, and then click **Nios II <*version*> Software Build Tools for Eclipse**. The **Workspace Launcher** dialog box appears.

2. Select a workspace folder by browsing to the video and image processing example design zip file software directory and specifying a workspace subdirectory. For example, **D:\VIP\vip_example_design_3c120_v100_revA\software\workspace**.
3. Click **OK** to create a new workspace.
4. On the File menu, point to **New**, and then click **Nios II Application and BSP from Template**. The **Nios II Application and BSP from Template** dialog box appears.
5. From the **SOPC Information File name** box, browse to the **system.sopcinfo** file. The Nios II SBT for Eclipse fills in **CPU name** with the processor name found in the **.sopcinfo** file.
6. In the **Project name** box, type `vip_control`.
7. Select **Blank Project** from the **Templates** list, as shown in Figure 29, and then click **Next**.

Figure 29. Nios II Application and BSP from Template Dialog Box


Nios II Application and BSP from Template

Nios II Software Examples

Create a new application and board support package based on a software example template

Target hardware information

SOPC Information File name: D:\acds91\nios2eds\workspace\system.sopcinfo

CPU name: cpu

Application project

Project name: vip_control

☑ Use default location

Project location: D:\acds91\nios2eds\workspace\software\vip_control

Project template

Templates: Blank Project, Board Diagnostics, Count Binary, Hello Freestanding, Hello MicroC/OS-II, Hello World, Hello World Small, Memory Test, Simple Socket Server, Web Server

Template description:

Blank Project creates an empty project to which you can add your code.

For details, click Finish to create the project and refer to the readme.txt file in the project directory.

The BSP for this template is based on the Altera HAL operating system. To use a BSP based on a different operating system, click Next and select the BSP from the BSP projects list.

For information about how this software example relates to Nios II hardware design examples, refer to the Design Examples page of the Nios II documentation available with your installation at: <installation_directory>/nios2eds/documents/index.htm.

8. Select **Create a new BSP project based on the application project template**.
9. Accept the default project name, `vip_control_bsp`.
10. Ensure that **Use default location** is turned on.

11. Click **Finish** to create the application and the BSP based on the **system.sopcinfo** file.
12. After the BSP generates, the **vip_control** and **vip_control_bsp** projects appear in the Project Explorer view. Outside of the Eclipse environment, open a new file explorer window, and navigate to the **software\source** directory in your VIP example design zip file, for example, **D:\VIP\vip_example_design_3c120_v100_revA\software\source**.
13. Select all 23 C++ VIP IP files and drag them onto **vip_control** in the Eclipse Project Explorer view to include them in the project. Ensure that you drag the files into the **vip_control** project, and not the **vip_control_bsp** project.

Building the Software in the Nios II SBT for Eclipse

Check that the application project (**vip_control**) and system library project (**vip_control_bsp**) are shown in the workspace.

1. Before building the **vip_control_bsp** project, reduce memory footprint by performing the following steps:
   a. Right-click **vip_control_bsp** in the Project Explorer view, and click **Properties**. The **Properties for vip_control_bsp** dialog box appears.
   b. Click **Nios II BSP Properties**.
   c. Turn on **Reduced device drivers** and **Small C library**, and ensure that **Support C++** is turned on (this is the default).
   d. Click **OK**. The **vip_control_bsp** project regenerates.
2. Right-click **vip_control** in the Project Explorer view, point to **Run as**, and then click **Nios II Hardware**. The Nios II SBT for Eclipse compiles the **vip_control_bsp** and **vip_control** software projects and builds the **vip_control.elf** software executable. After the build stage completes, the **Run Configurations** dialog box appears.
3. Verify that **vip_control** is selected in the **Project name** box, and the newly created **vip_control.elf** file is listed in the **ELF file name** box.
4. Load the **vip_top.sof** file to program the development board with hardware.
5. Click the **Target Connection** tab, and then click **Refresh Connections**.
6. Verify that the USB-Blaster™ and EP3C120 configuration device appear under **Connections**.
7. Click **Run**. The following appears in the Nios II Console output:

   ```
   F0 width x height= 720 x 244
   F1 width x height= 720 x 243
   ```

   The video appears on the output monitor.

Debugging the Application Source Code

To debug the application source code, perform the following steps:

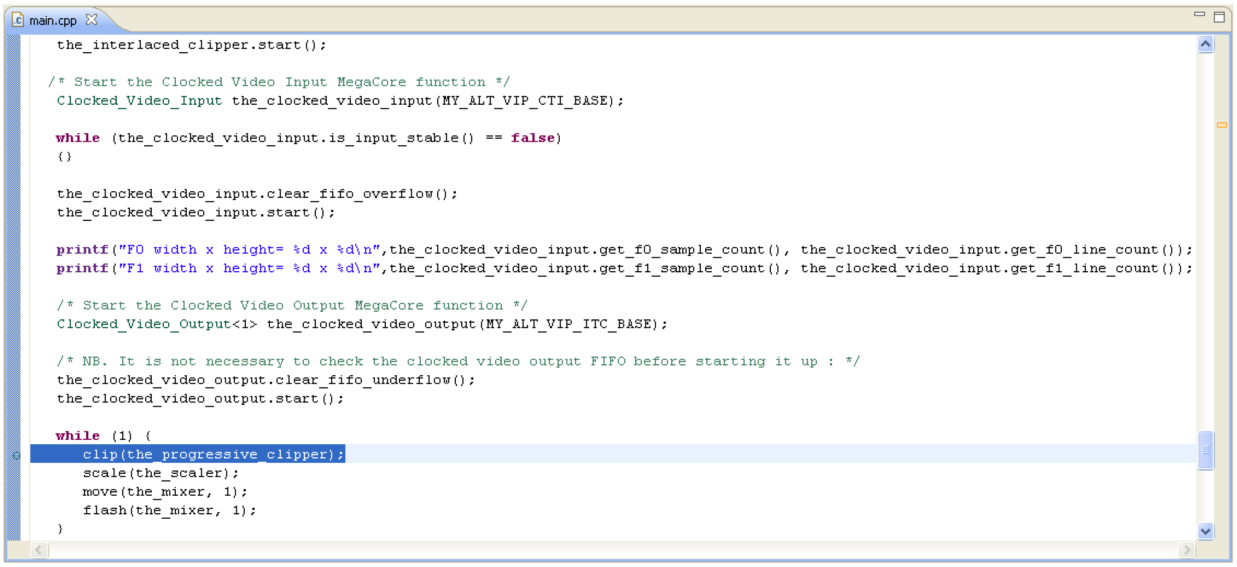
1. Right-click **vip_control**, point to **Debug As**, and then click **Nios II Hardware**.

2. A message asking you if you want to switch perspective might appear. Clicking **Yes** switches to the debug perspective view. The VIP hardware stops manipulating the video, and the program pauses at the start of the **main()** function.
3. As shown in Figure 30, insert a breakpoint by clicking in the margin next to the following line:

   ```
   clip(the_progressive_clipper);
   ```

   then right-click and click **toggle breakpoint**.

Figure 30. Inserting a Breakpoint in the Nios II Debug Window



```
   the_interlaced_clipper.start();

  /* Start the Clocked Video Input MegaCore function */
   Clocked_Video_Input the_clocked_video_input(MY_ALT_VIP_CTI_BASE);

   while (the_clocked_video_input.is_input_stable() == false)
   {}

   the_clocked_video_input.clear_fifo_overflow();
   the_clocked_video_input.start();

   printf("F0 width x height= %d x %d\n",the_clocked_video_input.get_f0_sample_count(), the_clocked_video_input.get_f0_line_count());
   printf("F1 width x height= %d x %d\n",the_clocked_video_input.get_f1_sample_count(), the_clocked_video_input.get_f1_line_count());

   /* Start the Clocked Video Output MegaCore function */
   Clocked_Video_Output<1> the_clocked_video_output(MY_ALT_VIP_ITC_BASE);

   /* NB. It is not necessary to check the clocked video output FIFO before starting it up : */
   the_clocked_video_output.clear_fifo_underflow();
   the_clocked_video_output.start();

   while (1) {
      clip(the_progressive_clipper);
      scale(the_scaler);
      move(the_mixer, 1);
      flash(the_mixer, 1);
   }
```

4. Click **Resume** to run the initialization code. The program breaks at the `clip(the_progressive_clipper);` line.
5. Click **Resume** again and notice the window dynamically clip, scale move and flash on the screen. The program breaks at the `clip(the_progressive_clipper);` line again.

   In a debug session, you can perform many debug operations including breakpoint insertion, viewing registers, viewing memory data, viewing variables, and single stepping instructions.

6. To end the debug session, click **Terminate**.

For more information about running a Nios II debug session, refer to “Getting Started” in the *Getting Started with the Graphical User Interface* chapter of the *Nios II Software Developer’s Handbook*.

Set Up the Hardware and Configure the FPGA

'This section describes how to set up the Cyclone III video Development Board, configure the Cyclone III device, and download the Nios II control program.

Quick start instructions are included in the top-level **vip_top.v** file.

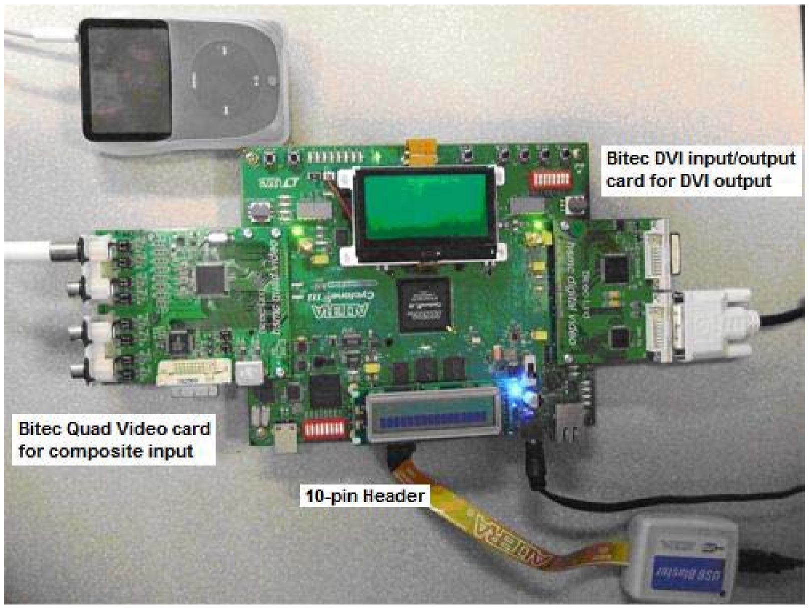
Set Up the Cyclone III Video Development Platform

The example design can be run with the Cyclone III Video Development Kit, and requires the following components:

- Cyclone III 3C120 Host Development Board
- Bitec HSMC Quad Video input daughtercard
- Bitec HSMC DVI input/output daughtercard
- Any NTSC video source with composite output and composite cable (for example: DVD player, camera, or iPod)
- A monitor or display with a DVI interface supporting 1024×768 resolution
- One DVI cable to connect the DVI output to the monitor
- Altera USB Blaster cable

Figure 31 shows the hardware setup.

Figure 31. Cyclone III Development Board Hardware Setup

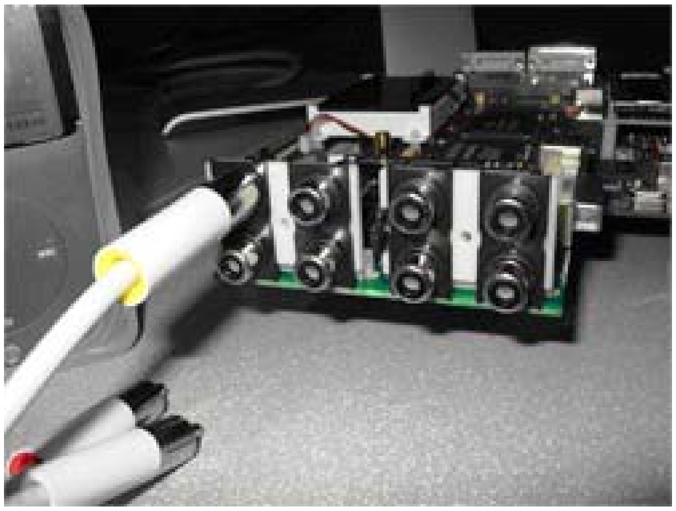

Bitec DVI input/output card for DVI output
Bitec Quad Video card for composite input
10-pin Header

To set up the Cyclone III development board, perform the following steps:

1. Disconnect the power cable from the Cyclone III development board.
2. Connect one end of the USB Blaster cable to the USB port on your PC.

3. Connect the other end of the USB Blaster cable to the 10-pin header labeled (J3) on the Cyclone III development board.
4. Connect the Bitec HSMC Quad Video input daughtercard to HSMC interface port B (J9) on the Cyclone III development board.
5. Connect the video source composite input cable to the top left input connector on the Bitec Quad Video input daughtercard as shown in Figure 32.

Figure 32. Composite Video Connection to Quad Video Daughtercard



6. Connect the Bitec HSMC DVI input/output video daughtercard to the HSMC interface port (J8) on the Cyclone III development board.
7. Connect one end of the DVI cable to a DVI monitor, capable of 1024×768 @ 60Hz. Connect the other end to the DVI-TX connector on the Bitec DVI input/output daughtercard.
8. Reconnect the power cable to the Cyclone III development board.

For details about installing the USB Blaster software driver on the host PC (located at <***quartus_install_dir***>**\drivers\usb-blaster**), refer to the *USB-Blaster Download Cable User Guide*.

For details about the Cyclone III DSP development board, refer to the *Cyclone III Development Board Reference Manual*.

Configure the Cyclone III Device

To configure the Cyclone III device, download the **vip_top.sof** file to the development board by following these steps:

1. On the Tools menu in the Quartus II software, click **Programmer**.
2. Click **Save As** on the File menu.
3. Navigate to the VIP example zip file design directory, for example, **D:\VIP\vip_example_design_3c120_v100_revA**, and in the **Save As** dialog box, type `vip_top.cdf` in the **File Name** box.

4. In the **Save as type** list, select **Chain Description File**.
5. Click **Save**.
6. In the Programmer window, select **JTAG** in the **Mode** list.
7. Click **Hardware Setup** to configure the programming hardware. The **Hardware Setup** dialog box appears.
8. In the **Hardware** column, click **USB Blaster**.
9. Click **Close** to exit the **Hardware Setup** dialog box.
10. Click **Add file**, browse to the VIP example zip file design directory and click **vip_top.sof**.
11. Turn on **Program/Configure** for **vip_top.sof**.
12. Click **Start**.

    The Programmer begins to download the configuration data to the FPGA. The **Progress** field displays the percentage of data that is downloaded. A message appears when the configuration is complete.

☞ If you are not using a licensed version of the Video and Image Processing Suite, a message appears indicating that you are running a time-limited configuration file on your target hardware.

The video appears on the screen when the control code has enabled the video processing pipe.

Download the Nios II Control Program

You can download the Nios II program to the FPGA in one of two ways:

- From a prebuilt binary object file via the Nios II command shell—This flow is recommended to quickly see the VIP demo working on the hardware.
- Building the binary from the Nios II SBT for Eclipse—This flow is recommended to understand the details of how the software is built.

Download vip_control.elf From a Command Shell

1. Start a Nios II Command Shell (On the Windows Start menu, point to **Programs**, then **Altera**, then **Nios II EDS** <*version*>, and click **Nios II** <*version*> **Command Shell**).
2. Change the directory to <*example design install*> **/vip_example_design_3c120_**<*version*>.
3. Download the **vip_control.elf** file and start the Nios II terminal window to display print output using the command:

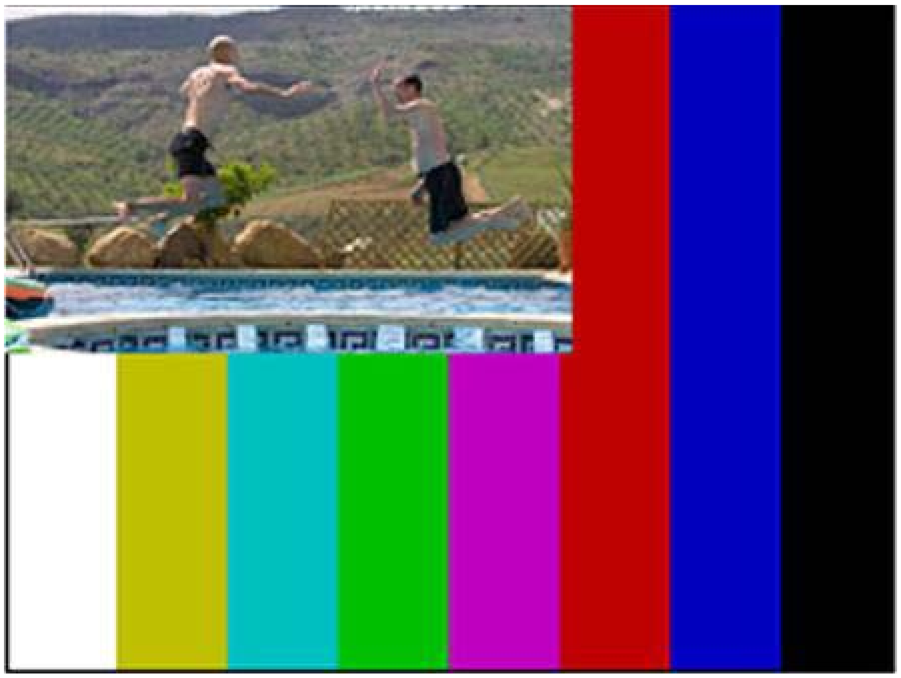
```
nios2-download -g -r vip_control.elf; nios2-terminal
```

   Verify that the resolutions of the input video fields (F0 and F1) are displayed in the Nios II terminal window, and that a video stream appears on the screen, dynamically clipping, scaling, flashing and changing location relative to a test pattern background.

Build and Load a New vip_control.elf From the Nios II SBT for Eclipse

1. Follow the steps described in “Review the Example Design” on page 11. This procedure builds and programs the Nios II processor with a new **.elf** file from the source code and the Qsys description of the hardware.
2. Verify that a video stream appears on the screen, dynamically clipping, scaling, flashing and changing location relative to a test pattern background. (Figure 33).

Figure 33. Video Stream Output



You can rebuild, edit, and debug the control code in the Nios II SBT for Eclipse with the software API classes provided to access the Video and Image Processing Suite functions. For more information, refer to “Nios II Software” on page 36.

You have successfully completed the Video and Image Processing example design demonstration.

Extending the Example Design

Many new video innovations, such as HDTV and digital cinema, involve video and image processing and this technology's rapid evolution. Leaps forward in image capture and display resolutions, advanced compression techniques, and video intelligence are the driving forces behind the technological innovation.

The move from standard definition (SD) to high definition (HD) represents a six times increase in data processing requirements. Video surveillance is also moving from common intermediate format (CIF) (352×288) to D1 format (704×576) as a standard requirement, with some industrial cameras moving to HD at 1280×720. Military surveillance, medical imaging, and machine vision applications are also moving to very high resolution images.

With expanding resolutions, there is a need for high performance while keeping architectures flexible to allow for quick upgradeability.

All of the processing functions within the Altera Video and Image Processing Suite are configurable to allow you to satisfy common performance and cost requirements over a wide range of SD and HD resolutions. This section describes examples of some of the many video systems that could be constructed using the Video and Image Processing Suite MegaCore functions.

Single Video Stream Input

The example design is a good starting point for developing single input or multiple input video stream applications. The example design demonstrates how to construct a video data path that performs deinterlacing, clipping, chroma resampling, color space conversion, mixing and scaling.

The Altera Video and Image Processing MegaCore functions all have common data and control interfaces. These common interfaces, and open protocol for transmitting video, make it easy to extend the data path in the example design to perform additional video processing functions. These can include other Video and Image Processing Suite functions, such as 2D finite impulse response (FIR) filtering, 2D median filtering, or gamma correction, as well as your own video processing components.

The example design also demonstrates how to use a Nios II processor to configure and control the video interface chips, clocked video input, clocked video output, frame buffer, and image mixer. The standardization on Avalon-MM control interface ports makes the Nios II processor a convenient choice for run-time control, enabling a rapid development cycle in a software environment.

As well as adding new functions, you can enable the run-time control interfaces on video processing functions, such as the Test Pattern Generator and Color Space Converter. These Avalon-MM slave control ports can be enabled from the MegaWizard interfaces. Once the hardware system is regenerated in Qsys and recompiled in the Quartus II software, control of the system is transitioned into the software domain.

The MegaCore function register maps provide both control and visibility of the data flowing in the data path. For example, you can perform the following:

- Perform real-time changes to the size of the test pattern stream.
- Update the scaling coefficients to improve image quality at run-time.
- Change the Color Space Converter coefficients at run-time.
- Track the number of frames dropped or repeated by the frame buffer due to data rate transitions.
- Monitor the number of words in the dual clock FIFO of the Clocked Video Input and Clocked Video Output MegaCore functions, to track overflow and underflow or peak FIFO usage.

Multiple Video Channel Input

Video systems frequently contain multiple video streams, which process in parallel and are synchronized. The Altera video and image processing design framework is suitable for constructing video systems with small numbers of channels (such as converter boxes) up to systems with hundreds of video streams (such as switchers and multiviewers).

The video up conversion example design is a good starting point for developing multiple video stream input applications.

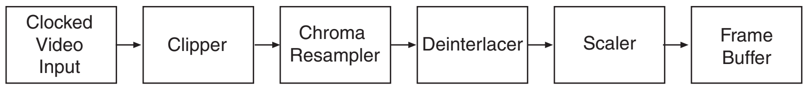
Using the 3C120 Video Development Kit, you can easily extend the example design to support four composite or S-Video video stream inputs via the Bitec HSMC Quad Video daughtercard, as well as the DVI input on the Bitec HSMC DVI input/output daughtercard.

For convenience, the Nios II software is provided with the design for you to configure the TVP5154 chip on the Bitec HSMC Quad Video input daughtercard for four composite inputs.

The following steps provide guidelines on how to extend the example design to support additional composite input video streams:

1. In Qsys, create an additional video processing data pipeline to input into the mixer, with the processing functions shown in Figure 34.

Figure 34. Qsys Processing Functions



You must ensure that the video functions process data at the required rate to avoid overflow at the input of the system and underflow at the output. When adding functions to a system that connect to external memory, such as a Deinterlacer, ensure that there is sufficient external memory bandwidth available.

2. Increase the number of inputs to the Alpha Blending Mixer.
3. Connect the new data pipeline to the new mixer input.
4. Regenerate the Qsys system.
5. Instantiate the new Qsys symbol in the top-level Quartus II **.v** file, add the appropriate Bitec Quad Video card input pins (the project already contains the correct assignments), connect the exported wires from the Qsys system to the new input pins, save the **.v** file, and recompile the Quartus II project.
6. Update the software project to initialize and control the mixer.

There are many challenges associated with multiple input stream systems, such as achieving the required level of external memory bandwidth, often with a shared DDR external memory; synchronizing multiple streams of video at different data rates; or processing at HD data rates. The video design framework is being aggressively developed to simplify such system and design challenges.

Other Video Interface Standards

The example design provides a framework to develop video systems with NTSC or PAL video input and DVI video output. This example requires appropriate hardware ports on the video development board, analog-to-digital conversion and FPGA hardware interface components.

A variety of different hardware interface standards and protocols are seen in applications throughout the broadcast, consumer, automotive, surveillance, medical, and document imaging markets. These include high definition multimedia interface (HDMI), component, S-Video or Y/C (with separate luminance and color signals), SDI, high definition SDI (HD-SDI) and 3GBit/s SDI (3G SDI).

The programmable nature of the FPGA lends itself well to migrating systems that support different interfaces. The example design could be used as a framework for development of such FPGA hardware interface components. This can be achieved by reparameterizing the Clocked Video Input and Clocked Video Output MegaCore functions in the Qsys system to match the video format and adapting the top-level design with the appropriate video interface and/or pin assignments.

For example, video systems with SDI video input and output ports using the Stratix® II GX board and the Altera SDI/HD-SDI MegaCore function, can be developed by applying the same video framework methodology as employed to this example design.

Conclusion

The Video and Image Processing example design demonstrates a reusable and flexible video framework for rapid development of video and image processing designs. Using standard open interfaces and protocols throughout the system allows you to build further applications, by reusing parameterizable IP from the Altera IP library or by adding your own IP to the framework.

System control code is implemented in software running on a Nios II processor, providing a rapid edit debug cycle. A C++ class based software API provides an interface to the Video and Image Processing Suite MegaCore functions from the Nios II control software to facilitate software development.

The video framework does not preclude use of HDL to connect the IP components. However, the example design demonstrates that the Qsys environment significantly accelerates system design by:

- Automatic generation of an application-specific switch fabric including arbitration scheme
- Providing an abstracted view of the video system
- Detecting and displaying Altera and user IP in an immediately accessible form

Revision History

Table 2 shows the revision history for the *AN-427: Video and Image Processing Example Design* application note.

Table 2. AN-427 Revision History

Version	Date	Change Summary
9.0	June 2011	Updated for Qsys version 11.0, new MegaWizard interface, new Deinterlacer II MegaCore and Scaler II MegaCore functions in the Video and Image Processing Suite, and minor changes to the example design.
8.1	July 2010	Updated for SOPC Builder version 10.0, new MegaWizard interface, new Interlacer MegaCore function in the Video and Image Processing Suite, and minor changes to the example design.
8.0	November 2009	Updated for Nios II SBT for Eclipse, SOPC Builder version 9.1, Quartus II software version 9.1, new Video and Image Processing Suite MegaCore functions, and minor changes to the example design.
7.0	March 2009	Updated for Quartus version 9.0. The OpenCores I2C Master warning message errata is fixed and the **Clocked_Video_Input.hpp** status register API is corrected.
6.1	January 2009	Minor corrections.
6.0	December 2008	Updated for Quartus version 8.1. Revised design that uses run-time control of the Clipper, Deinterlacer, Scaler, and Frame Buffer MegaCore functions and supports PAL video as an alternative to NTSC.
5.0	July 2008	Updated for Quartus version 8.0. Completely revised design for using the Cyclone III Video Development kit with Bitec HSMC daughtercards. The example design is implemented by MegaCore functions instantiated in SOPC Builder with control provided by a Nios II processor.
4.0	October 2007	Updated for Quartus version 7.2. The design now uses DSP Builder Video Source and Video Sink blocks and the triple buffer block is replaced by a Frame Buffer MegaCore function. Removed obsolete troubleshooting issue “Compilation Fails in Analysis and Synthesis”.
3.0	May 2007	Updated for Quartus version 7.1.
2.0	December 2006	Updated for Quartus version 6.1.
1.2	July 2006	Added troubleshooting issues “Compilation Fails in Analysis and Synthesis” and “DDR2 Fails in Analysis and Synthesis or In Timing Analysis”.
1.1	July 2006	Updated algorithm used by the triple buffer block, effects of VGA starvation, description of the image stream frame counter block, and other minor edits.
1.0	June 2006	First release of this application note.